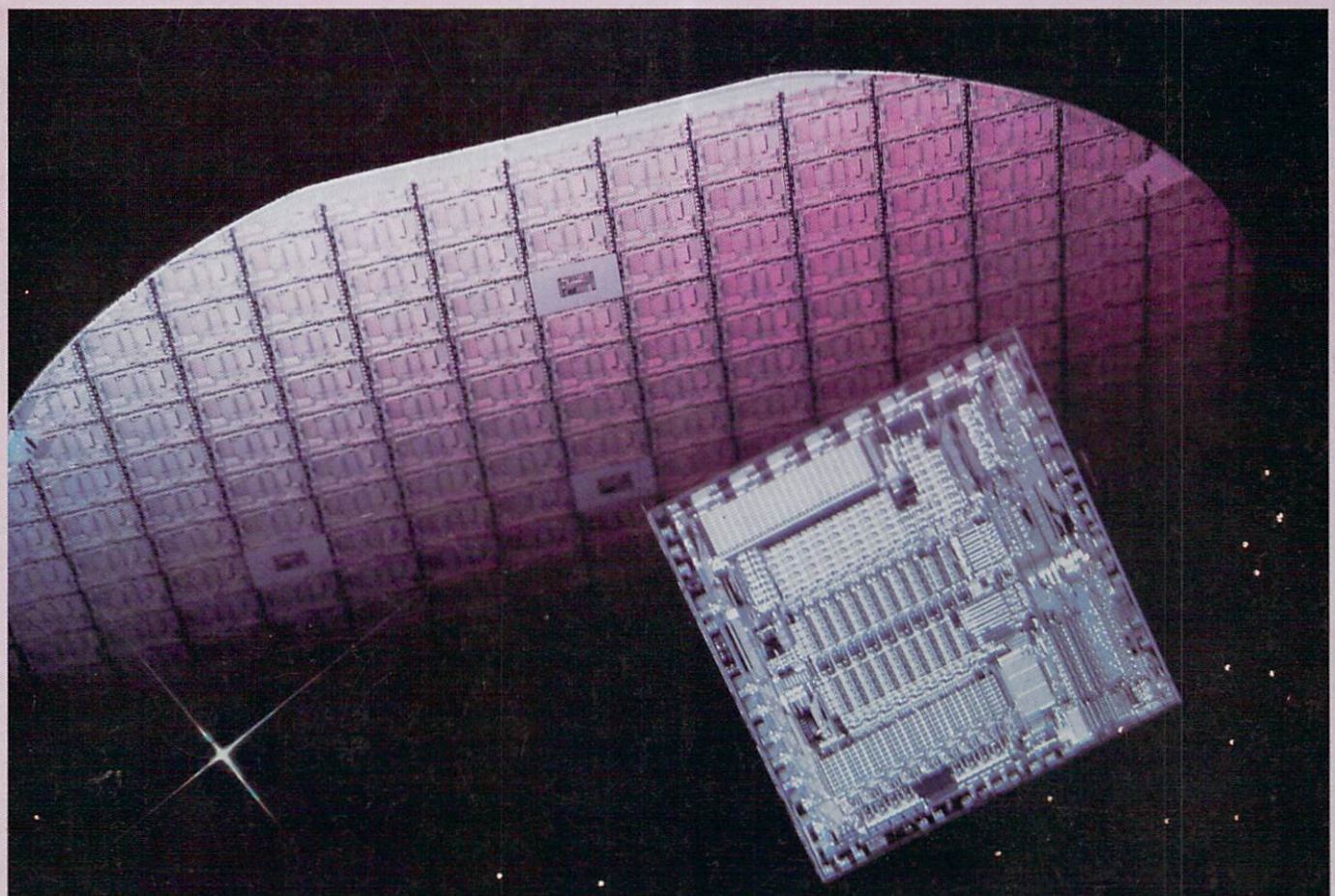
commodore
the microcomputer magazine

Volume 4, Number 6, Issue 27
$2.50 U.S.
$3.50 Canada
ISSN 0774-8724



Semiconductor Technology

How Chips Are Made

Happy 25th Birthday, Commodore!

Commodore Business Machines, Inc.
1200 Wilson Drive · West Chester, PA 19380

COMMODORE BOOKWARE

Hardware... Software... BOOKWARE

Commodore offers you complete support for your home computer. To complement your Commodore hardware and software, we have a full range of "teach yourself" programming and instructional books. From "create your own graphics and music" to extensive computer language programming, Commodore meets your computer needs.

The Commodore Software Encyclopedia is an indispensable guide for worldwide Commodore software.

From beginner to business professional Commodore Books are valuable additions to your computer library.

commodore COMPUTERS

First In Quality Software

Don't let price get in the way of owning a quality printer.

Adding a printer to your computer makes sense. But deciding which printer to add can be tricky. Do you settle for a printer with limited functions and an inexpensive price tag or buy a more versatile printer that costs more than your computer? Neither choice makes sense.

Here's a refreshing option—the new, compact STX-80 printer from Star Micronics. It's the under $200 printer that's whisper-quiet, prints 60 cps and is ready to run with most popular personal computers.

The STX-80 has deluxe features you would expect in higher priced models. It prints a full 80 columns of crisp, attractive characters with true descenders, foreign language characters and special symbols. It offers both finely detailed dot-addressable graphics and block graphics.

And, of course, the STX-80 comes with Star Micronics' 180 day warranty (90 days on the print element).

The STX-80 thermal printer from Star Micronics. It combines high performance with a very low price. So now, there is nothing in the way of owning a quality printer.

*Manufacturer's suggested retail price.

star™
micronics·inc

THE POWER BEHIND THE PRINTED WORD.

Computer Peripherals Division, 1120 Empire Central Place, Suite 216, Dallas, TX 75247 (214) 631-8560

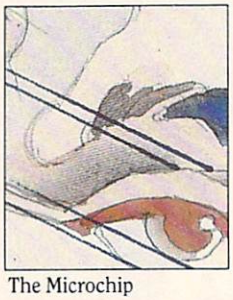
The new STX-80 printer for only $199.*

commodore
the microcomputer magazine

features

Volume 4, Number 6, Issue 27

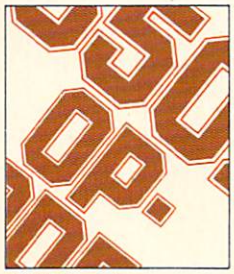

In the Chips


The Microchip


Bits and Pieces

departments

commodore
the microcomputer magazine


6502 Op-Codes

staff

Publishing Manager
Neil Harris
Editor
Diane LeBold
Technical Editor
Jim Gracely
Staff Writers
Jeff Bruette
Brooks Cooley
Mark Cornacchio
Barbara Karpinski
April Koppenhaver
Contributing Writers
David Bull
M.W. Caprio
James E. Darrough
Elizabeth Deal
Doris Dickenson
Andy Gamble
Jeff Hand
Donald Hassler
Len Lindsay
F.H. Shedd
Leslie Wood
Mark Zimmermann
Tom Zurkammer
Technical Staff
Barbara Karpinski
April Koppenhaver
Cyndie Merten
Sarah Meyer
Stephen Murri
Circulation Manager
John O'Brien
Circulation Assistant
Kathy Reigel
Advertising Coordinator
Sharon Steinhofer
Graphic Design
Neumann Greenberg Schlenker
King of Prussia, Pennsylvania
Cover Photo
Bob Emmott
Typography
Associates International, Inc.
Wilmington, Delaware
Printing
Volkmuth Printers
St. Cloud, Minnesota

ISBN 0-88731-005-2

Subscription Information: U.S. subscriber rate is $15.00 per year. Canadian subscriber rate is $20.00 per year. Overseas $25.00. To order phone 800-345-8112 (In Pennsylvania 800-662-2444).

Our 1984 Issues Continue to Bring You the Best Information Straight from the Source!

Commodore: the Microcomputer Magazine, Issue 28: More up-to-the-minute information for Commodore enthusiasts—programming tips; applications in education, business, the arts and the home; product reviews; user groups—and maybe a few surprises! Look for us in February.

Power/Play, Spring: On the lighter side of Commodore computing, *Power/Play* continues to bring you program listings, information on the Commodore Kids, learning at home, entertainment, home applications and more. We'll be out there in March with all the computing fun you can handle.

Key to Entering Program Listings

```
"[F1,F2,F3,F4,F5,F6,F7,F8]":F1,F2,F3,F4,
  F5,F6, F7 AND F8
"[POUND]":ENGLISH POUND
"[PI]"PI SYMBOL
"^":UP ARROW
"[HOME]":UNSHIFTED CLR/HOME
"[CLEAR]":SHIFTED CLR/HOME
"[RVS]":REVERSE ON
"[RVOFF]":REVERSE OFF
"[BLACK,WHITE,RED,CYAN,MAGENTA,GREEN,BLUE,
  YELLOW]" THE 8 CTRL KEY COLORS
"[ORANGE,BROWN,L. RED,GRAY 1,GRAY 2,L.
  GREEN,L. BLUE,GRAY 3]":THE 8
  COMMODORE KEY COLORS (ONLY ON THE 64)
GRAPHIC SYMBOLS WILL BE REPRESENTED AS
  EITHER THE LETTERS SHFT (SHIFT KEY) AND
  A KEY:"[SHFT Q,SHFT K,SHFT V,SHFT T,
  SHFT L]"
  OR THE LETTERS CMDR (COMMODORE KEY) AND
  A KEY:"[CMDR Q,CMDR H,CMDR S,CMDR N,
  CMDR O]"
IF A SYMBOL IS REPEATED, THE NUMBER OF
  REPITITIONS WILL BE DIRECTLY AFTER THE
  KEY AND BEFORE THE COMMA:"[SPACE3,
  SHFT S4,CMDR M2]"
```

commodore 64

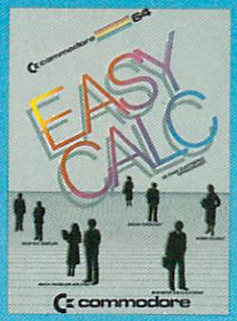
INDISPENSABLE SOFTWARE

For Your Most Important Computing Needs

Commodore is your best value in practical software—just take a look at the programs shown here—we've got everything from wordprocessing to business accounting, from electronic spreadsheets to computer graphics. Use the Software Selection Guide to find the programs which best meet your needs, then see your Commodore dealer!

EasyScript 64
Displays 764 lines × 240 characters. Prints to 130 columns. Works with EasySpell 64.

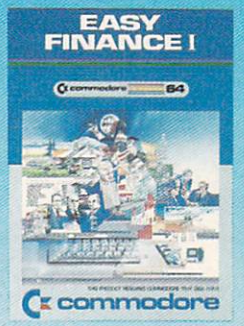
EasySpell 64
20,000 word Master Dictionary and automatic spelling checker. Works with EasyScript 64.



EasyCalc 64
Multiple electronic spreadsheet with color bar graph feature. 63 columns × 254 rows.

The Manager
Sophisticated database system with 4 built-in applications, or design your own. Text, formulas, graphics.

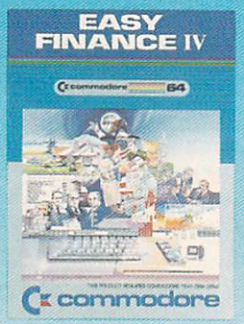
SuperExpander 64
21 special commands. Combine text with high resolution graphics. Music and game sounds.



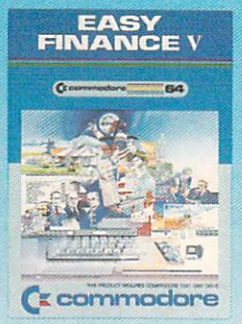
Easy Finance I—Loan Analysis
12 loan functions. Bar graph forecasting as well as calculation.

Easy Finance II—Basic Investment Analysis
16 stock investment functions. Investment bar graph.

Easy Finance III—Advanced Investment Analysis
16 capital investment functions. Bar graphs.



Easy Finance IV—Business Management
21 business management features. Bar graphs.



Easy Finance V—Statistics and Forecasting
Assess present/future sales trends with 9 statistics and forecasting functions.

Accounts Payable/ Checkwriting
11 functions. Automatic billing. 50 vendors/disk.

Accounts Receivable/Billing
11 billing functions. Printed statements.

General Ledger
8 general ledger options. Custom income statement, trial balances, reports.

Inventory Management
1000 inventory items. Full reports.

Payroll
24 different payroll functions. Integrated with G/L system.

SOFTWARE SELECTION GUIDE

APPLICATION	SOFTWARE
Budget/Calculation	EASYCALC 64
Business Accounting	ACCOUNTS PAYABLE/CHECKWRITING, ACCOUNTS RECEIVABLE/BILLING, GENERAL LEDGER, INVENTORY MANAGEMENT, PAYROLL
Business Management	EASYFINANCE IV—BUSINESS MANAGEMENT
Children's Programming	ZORTEK & THE MICROCHIPS
Cooking/Recipes	MICRO COOKBOOK
Data Base Management	THE MANAGER
Electronic Spreadsheet	EASYCALC 64
Filing/Recordkeeping	MAGIC DESK, THE MANAGER, INVENTORY MANAGEMENT
Financial Investments	EASYFINANCE II—BASIC INVESTMENT ANALYSIS, EASYFINANCE III—ADVANCED INVESTMENT ANALYSIS, FINANCIAL ADVISOR
Graphics/Sound	SUPEREXPANDER 64
Learn Programming	INTRODUCTION TO BASIC—PART 1
Loans/Mortgages	EASYFINANCE I—LOAN ANALYSIS, FINANCIAL ADVISOR
Mailing List	EASYMAIL 64
Music	MUSIC COMPOSER, MUSIC MACHINE
Programming Aids	SUPEREXPANDER 64, SCREEN EDITOR, ASSEMBLER 64
Reference Books	PROGRAMMERS REFERENCE GUIDE, SOFTWARE ENCYCLOPEDIA
Spelling Dictionary	EASYSPELL 64 (for use with EASYSCRIPT 64)
Statistics/Forecasting	EASYFINANCE V—STATISTICS & FORECASTING, EASYFINANCE IV—BUSINESS MANAGEMENT
Teacher's Aids	EASYLESSON/EASYQUIZ, LOGO, PILOT
Telecommunications	VICMODEM, AUTOMODEM, TERM 20/64, RS232 INTERFACE
Wordprocessing	EASYSCRIPT 64, MAGIC DESK, WORD MACHINE/NAME MACHINE

MAGIC DESK I-TYPE & FILE

Only Commodore brings you the magic of MAGIC DESK... the next generation of "user-friendly" software! Imagine using your computer to type, file and edit personal letters and papers—without learning any special commands! All MAGIC DESK commands are PICTURES. Just move the animated hand to the picture of the feature you want to use (like the TYPEWRITER) and you're ready to go. MAGIC DESK is the "ultimate" in friendly software!

Special "Help" Menus

Not only is MAGIC DESK easy to use... it's hard to make a mistake! Just press the COMMODORE key and one of several "help menus" appears to tell you exactly what to do next.

commodore COMPUTERS

First In Quality Software

commodore 64

THE BEST GAMES IN TOWN

Arcade Action Games

Pinball Spectacular: Real pinball action and thrills. Sound you won't believe. Chutes, lights, bumpers and more.

Supersmash: Raquetball arcade classic. 3 games in 1. Many skill levels keep the challenge alive.

Tooth Invaders: Reviewed by American Dental Association. Arcade action teaches good dental care. Beat D.K. at all 9 play levels.

NEW **Star Post:** Protect the Star Post from waves of invaders. 3 levels of skill. 99 levels of action.

NEW **Avenger:** Destroy attacking aliens with laser cannons. Classic arcade action. Multi-speed attacks.

Jupiter Lander: Space landing simulation. Horizontal/vertical thrust. Soft-land scoring. Wow! animation.

Radar Rat Race: Beat the maze. Eat all the cheese. Beware deadly cats/rats. Cartoon action fun for all ages.

NEW **Lemans:** Multi-obstacle road racing at its best. Arcade action and graphics. Night, water and divided highway hazards.

NEW **Star Ranger:** Fight your way through hoards of space enemies. Avoid asteroids and land safely. Superb graphics and space action.

NEW **Frogmaster:** Unique sports challenge. Train animals to play football and rugby. Over 100 variations. Play against, computer, friend or yourself.

Children's Series

Introduction to Basic I: Simple step-by-step instructions. Modular design. Practical BASIC applications as you learn.

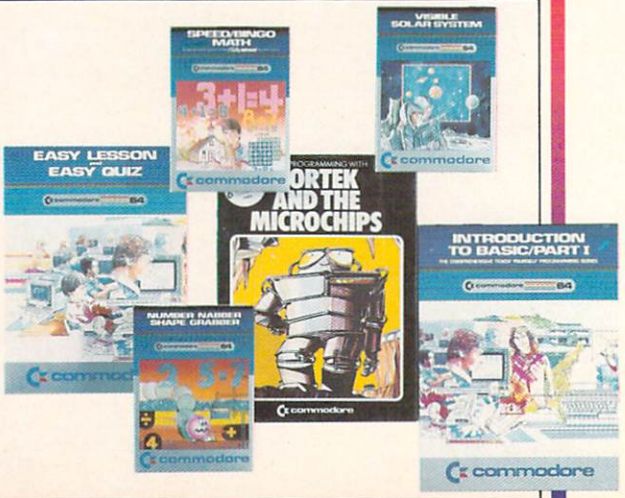
Zortek and the Microchips: Award winning program teaches children BASIC through games, graphics and stories.

Easy Lesson/Easy Quiz: Take the drudgery out of writing tests and quizzes. Answer keys provided. 7 categories per test.

NEW **Number Nabber/Shape Grabber:** 2 Learning games in 1. Build both math and object identification skills. Lively graphic and sound effects.

Visible Solar System: Fly the solar system. Land on planets. Calculate age and weight. Astronomy for home and school. Award winner.

Speed/Bingo Math: 2 games in 1 teach children 4 to 10 basic math skills. Beat the clock or your friends.



Bally Midway

NEW **Gorf:** 4 Space action games in 1. Fly your fighter defeat "The Empire". Multi-skill levels. IT TALKS! (with Magic Voice)

NEW **Wizard of WOR:** Fight your way through 30+ mazes. Defeat the Wizard and Worriors. Multi-skill. IT TALKS! (with Magic Voice.) Award winning conversion.

Seawolf: The classic battle at sea. Destroy PT Boats and Destroyers. Great graphics and sound.

Omega Race: Fast space race action. Many skill levels. Avoid deadly mines as you eliminate droid forces.

Clowns: Amazing action under the "Big Top". Help clowns "pop" balloons. Colorful acrobatics. Fun for all.

Kickman: Ride the unicycle and catch falling objects. Multi-skill levels. Tuneful sound. Watch out! Don't fall!

NEW **Blueprint:** Help J.J. build the "Ammo Machine". Parts are stored in a colorful maze of houses. Multi-skill and difficulty levels.

NEW **Lazarian:** 4 different screens. Multi-skill level space action. Rescue, evade obstacles and destroy a one-eyed leviathan.

Adventure Games

Zork I: Fantasy adventure in a dungeon. Find all the treasure and escape alive.

Zork II: This dungeon adventure dares you to find treasure and secret places and still survive.

Zork III: The ultimate dungeon test. Discover the Dungeon Master's secret purpose and come out alive.

Suspended: Awake in 500 years. Solve varied real and original puzzles to save the planet from total destruction.

Starcross: Travel through the mystery ship. Meet aliens friend and foe. Face the challenge of your destiny. Map of galaxy included.

Deadline: Find the murderer and solve the mystery all in 12 hours. Inspector casebook and evidence included.

Music Series

Music Machine: Play piano or organ melodies and percussion rhythms together. Music staff shows notes on screen. Vibrato, tempo and pitch controls.

Music Composer: Create, play and save your tunes easily. Simulates up to 9 instruments. Notes appear on screen. Play your keyboard like a piano.

commodore COMPUTERS

First In Quality Software

commodore

VIC 20

SUPER SOFTWARE SAVINGS

Bally Midway

Gorf: 4 Space action games in 1. Fly your fighter defeat "The Empire". Multi-skill level.

Seawolf: The classic battle at sea. Destroy PT Boats and Destroyers. Great graphics and sound.

Omega Race: Fast space race action. Many skill levels. Avoid deadly mines as you eliminate droid forces.

Clowns: Amazing action under the "Big Top". Help clowns "pop" balloons.

Lifestyle Series

Quizmaster: Write and give your own quizzes. Teach, revise, test and entertain.

Know Your Child's I.Q.: 3 Comprehensive tests. 100 questions. Auto and tamperproof scoring. Improve school test performance.

Know Your Own I.Q.: 4 I.Q. tests. 160 problems. Auto and tamperproof scoring. For hours of entertainment.

Know Your Personality: 3 In-depth personality tests. 450 questions. Auto scoring. Find your friends true feelings. For entertainment only.

Robert Carrier's Menu Planner: 120 meals and 20 wines start your menu data-base. Add your own recipes.

Children's Games

The Sky is Falling: Pre-school and elementary age children help Chicken Little. Builds hand-eye coordination.

Mole Attack: Bop the nasty moles as they stick heads out of burrows. Cartoon graphics. Multi-speed action.

Home Babysitter: Building blocks teach the alphabet. Common objects teach numbers to 20. Plus funny face maker.

Visible Solar System: Fly the solar system. Land on planets. Calculate age and weight. Astronomy for home and school. Award winner.

Speed/Bingo Math: 2 games in 1 teach children 4 to 10 basic math skills.

Business and Financial

Personal Finance: Four programs in one. Track expenses. Spending analysis. Budgets and deductibles.

Simplicalc: Electronic spreadsheet. 1200 entries. Design/repeat formulas and worksheets.

VIC Writer: Wordprocessing made simple. From 45 lines unexpanded to 1207 lines of text with 16K RAM PACK.

Money Decisions I: 7 Loan analysis functions. Principle, regular/last payment. Balance. Time period. Interest. Variable rate loan.

Money Decisions II: 9 investment functions. Future/Initial/Minimum Investment. Regular deposit/withdraw. Interest. Annuity Continuous compounding.

Educational Programs

Introduction to Basic I & II: Simple step-by-step instructions. Practical BASIC applications.

Zortek and the Microchips: Award winning program teaches children BASIC through games, graphics and stories.

Waterloo Basic: The original course in VIC BASIC.

Chopper Math: Challenging helicopter landing game that teaches math basics.

Easy Type: Learn touch-typing the easy way.

Adventure Games

Adventureland: Fantasy adventures challenge you to get all the treasure and escape alive.

Pirate Cove: Find the long lost treasure of pirate John Silver. Uncover clues while battling foes.

Atomic Mission: Save the nuclear powerplant from destruction. Piece clues together—solve the mystery.

The Count: Make your way through the dungeon, collect treasure and kill Count Dracula.

Voodoo Castle: Find the Count of Monte Cristo and remove the deadly curse.

Arcade Action Games

VIC Avenger: Destroy attacking aliens with laser cannons. Classic arcade action. Multi-speed attacks.

Super Alien: Trapped in an alien maze, your only defense is an alien buster. Hi-speed action.

Superslot: Vegas and Atlantic City casinos come home. Real slot machine action, graphics and sound.

Jupiter Lander: Space landing simulation. Horizontal/vertical thrust. Soft-land scoring.

Draw Poker: Casino style action. Betting. Sound effects.

Road Race: Night driving challenges you to the max. 4-speed shift. Stay on course. Don't overheat.

Radar Rat Race: Beat the maze. Eat all the cheese. Beware deadly cats/rats.

Raid on Ft. Knox: Sneak gold bars past deadly panthers and back to the hideout before time is up.

Pinball Spectacular: Space action and pinball thrills combined. Lights, bumpers, and special skill bonuses.

Sargon II Chess: Challenging chess strategy classic. Multi-skill levels from beginner to advanced.

Supersmash: Raquetball arcade classic. 3 games in 1. Many skill levels keep the challenge alive.

Cosmic Cruncher: Make your way through the Milky Way. 11 levels of play. Over 300 color/maze combinations.

Money Wars: Grab the money and run. 3 brick barricades are your protection as you dodge deadly bullets.

Tooth Invaders: Arcade action teaches good dental care. Beat D.K. at all 9 play levels.

Star Post: Protect the Star Post from waves of invaders. 3 levels of skill. 99 levels of action.

commodore COMPUTERS

First In Quality Software

commodore 64

MAGIC DESK I

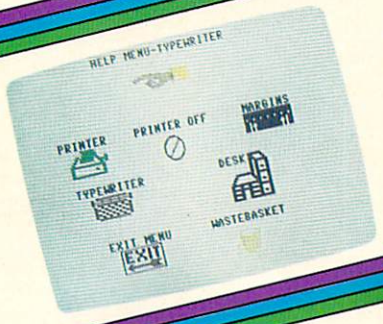
TYPE AND FILE

**Only from Commodore—
the excitement
and simplicity of
Magic Desk**

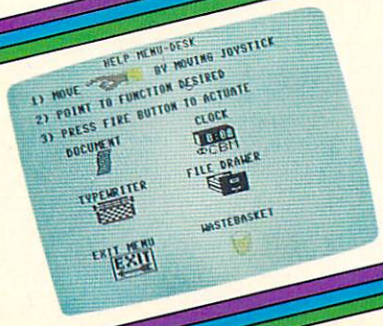
Only Commodore brings you the magic of MAGIC DESK... the next generation of "user friendly" software! Imagine using your computer to type, file and edit personal letters and papers *without learning any special commands!* All MAGIC DESK commands are PICTURES. Just move the animated hand to the picture of the feature you want to use (like the TYPEWRITER) and you're ready to go.

The MAGIC DESK Typewriter works just like a real ELECTRIC TYPEWRITER ... and it's COMPUTERIZED. All the filing is *electronic*. Excellent sound effects and screen animation make typing fun, whether you're typing letters, reports or memos ... and the built-in filing feature makes MAGIC DESK useful for keeping names and addresses, home inventory lists, insurance information and more.

Your COMMODORE 64, COMMODORE DISK DRIVE and MAGIC DESK are an unbeatable combination. Filing operations are automatically linked to your Commodore disk drive—but you don't have to know any commands—just "file" the pages you type in the file cabinet and your text is automatically saved on diskette. There are 3 file drawers with 10 file folders in each drawer and 10 pages in each folder.



To PRINT a page you've typed, just "point" at the picture of the printer and your pages are automatically printed on your COMMODORE PRINTER or PRINTER/PLOTTER. If you want to erase what you've typed, the WASTE-BASKET under the desk lets you "throw away" pages. There's even a DIGITAL CLOCK which helps you keep track of time while you're typing.



Not only is MAGIC DESK easy to use ... it's hard to make a mistake! Just press the COMMODORE key and one of several "help" menus" appears to tell you exactly what to do next. Special messages show you how the various picture commands work and help you when you make a mistake. Help messages also show you how to use the printer, filing cabinet, digital clock and wastebasket.

commodore COMPUTERS

First In Quality Software

Commodore Celebrates Its Silver Anniversary

The big news this month is the "World of Commodore" show in Toronto, Commodore's birthplace, in honor of the company's twenty-fifth birthday. Since many readers have asked us to publish a history of the company we thought this was the opportune time. You'll find it on page 17. We hope it answers a lot of the questions that may have been rattling around in your mind about the "upstart" company that ended up as number one in this highly competitive industry.

Coincidentally, this is also the issue that takes you through MOS Technology, Inc., the subsidiary of Commodore that produces all our microchips. Since it was the inexpensive 6500 family of microprocessors developed at MOS Technology that brought you the first truly affordable microcomputers—culminating in the VIC 20 and Commodore 64—and made Commodore one of the outstanding growth companies in the United States and around the world, it seems appropriate that we should take a look in this anniversary issue at just what they do over there in Norristown. Besides which, the process of chip making is pretty interesting in and of itself.

I have to admit that in researching the article about microchips, I learned a whole lot. Our technical editor, Jim Gracely, who already knows all that hardware-type stuff—or almost, anyway—seemed to be getting a kick out of listening to my great "revelations", and dutifully read my final story to make sure I hadn't said anything stupid. So if I've said anything stupid, you can blame him. Only don't tell him I said that.

I did all my philosophizing about the impending arrival of the year 1984 in *Power/Play* last month, so I won't go through it again. (It's hard to ignore the fact, nevertheless, that there it is, looming on the horizon.) What fascinates me, though, is how far computers have come since Orwell wrote that book and how our attitudes have changed toward them. In 1948, which is when *1984* appeared, we had the leviathan ENIAC. OK, it was primitive. The astounding thing is that its creators estimated that it would take only about *four* such "super calculators" to take care of all the computational needs of the whole U.S.A. And not many more to take care of the world. (No wonder Orwell was paranoid about computers.) Their naivete (or was it hubris?) makes you wonder, doesn't it? What assumptions are we operating under today that in thirty years or so will seem as silly as that one?

Commodore as a company has had an outstanding year. We in the publishing department have had an equally good one, thanks to the many readers who have taken the time to send us articles and letters, an unbeatable design team who revamped our whole "look" to make us (in my humble opinion) the most attractive computer magazine out there, and the people in Commodore's management who gave us the support we needed to make the magazines more accurate, prettier and generally more useful to the crucial person at the other end—you, the Commodore computer user.

All of us here at the big C would like to wish all of you out there at the other end a holiday season full of joy and a new year that brings you peace and prosperity. See you in February. C

—Diane LeBold
Editor

letters

Get the Best Picture From Your Monitor

To the Editor:

Although an attractive feature of the Commodore 64 computer is the ability to attach it directly to a television set, many business, educational, and scientific uses are better served with the purchase of a separate video monitor. The Commodore 64 is inherently capable of producing an excellent picture on such a monitor, but if my own experience is typical, many individuals are getting short-changed in this area. The major problem is that the cables sold for attaching monitors to the Commodore 64 can be wired in several different ways. They all produce a picture, but the disparity in picture quality is enormous.

In order to achieve optimal picture quality with a monochrome monitor (green, amber, or black/white), one must be at least superficially acquainted with the operation of the audio/video output socket. In addition to the audio signal, this output socket provides two types of video signals: one, called the *luminance* signal, provides brightness information while the other, termed the *video* signal, provides color information. A friend of mine who purchased a monitor cable for his Commodore 64 at a local computer store was given a *video* cable which, in spite of its misleading name, provides a rather fuzzy picture when used with a monochrome monitor. I purchased a cable at a second computer store, and was given one in which the *video* and *luminance* signals were mixed. The result is a better picture than with a video cable alone, but an even sharper picture is obtained when one uses a *luminance* cable in which only the luminance signal is employed.

The situation is even more complicated when using a color monitor. The Commodore 1701 color monitor I purchased comes with a two-jack cable, which plugs into the video and audio sockets on the front of the monitor. The manual states that a better picture can be obtained by using the three sockets in the back (audio, luma, and chroma), but warns that the computer must have an eight-pin audio/video socket in order to attach the monitor in this way. I have found, however, that in spite of the fact that the audio/video socket on my Commodore 64 is a five-pin socket, it can be attached by an appropriate cable to the sockets on the back of the monitor. The result is an enormous improvement in picture quality. Many color combinations that were virtually unreadable when the monitor was attached through its video and audio sockets on the front now give me excellent picture quality. The secret is to wire the luminance signal (pin #1) to the luma socket on the back of the monitor, the video signal (pin #4) to the chroma socket on the back of the monitor, and the audio-out signal (pin #3) to the audio socket on the back of the monitor. If you cannot find such a cable locally, they are relatively easy to wire or have wired for you. If you make your own cable, the location of these three pins is given in the Commodore 64 user's guide (page 142). You should note, however, that the diagram in the manual is for the audio-video socket on the back of the computer, and that the pins on the cable plug will therefore be the mirror image.

Armed with the above information, one can attain excellent picture quality with the Commodore 64 using either a monochrome or color monitor. It is unfortunate that many stores, at least in our area, are not knowledgeable about these facts and even demonstrate floor models of the Commodore 64 improperly wired to a video monitor.

Lewis J. Kleinsmith
Ann Arbor, Michigan

Calculating Folds

To the Editor:

I realize that Jim Gracely's program on page 58 of the June/July issue of Commodore was simply for illustration of the Stop-Print method of debugging. However, just in case someone is actually looking for a program to calculate new length after a number of folds, the following is much simpler and more direct and would probably never require debugging:

```
10 INPUT "LENGTH IN
   FEET"; A
20 INPUT "NUMBER OF
   FOLDS"; B
30 L=A/2↑B
```

letters

40 PRINT "NEW LENGTH IS" L "FEET."

Even though I am an absolute beginner at computing, I enjoy your magazine very much.

Ralph G. Smith
Phoenix, Arizona

Aligning Decimals

To the Editor,

I would like to suggest a fairly easy approach to the problem of aligning the decimal points in a column of numbers as an alternative to the more complete treatment in "Dollars and Cents Make Sense" on page 90 of Volume 4, Number 3.

The demonstration program below precedes the number with a sufficient number of spaces to align the decimal points of all but E-format numbers. The guts of this routine are lines 20 and 25, which may be structured as a subroutine for more general use. This routine will not add zeros to fill out places beyond the decimal point, nor will it include a decimal point for an integer.

```
10 INPUT Z
15 PRINT TAB(10);"{CU}";
20 FOR E=6TO0STEP-1:IF
   ABS(Z)<INT(10^E)THEN
   PRINT" ";:NEXT
25 IF Z<>0 THEN PRINT"";
30 PRINT Z
40 GOTO 10
```

Alvin H. Bachman
Spring Valley, New York

More on Enhanced Remarks

To the Editor:

Your article on enhanced remarks (May, 1983) sounded like a great idea. I was curious to see if the routine could be adjusted to run on the Commodore 64. After looking the program over I found that there were only a few lines that had to be adjusted.

The first thing that must be changed is the start of BASIC. BASIC on the Commodore 64 starts at memory location 2048. Line 50000 should read:

```
FORX=2048 TO 50000
```

Next, the code to the printer to print expanded characters has to be changed. The code for the Commodore 64 is CHR$(14). Line 50030 should read:

```
IF PEEK(X)=143 AND
PEEK(X+1)=255 THEN
POKE X+1,14
```

This slightly adjusted routine will give program listings enhanced remarks and improved readability on the Commodore 64. One more note. On printers with a special listing mode for cursor control symbols, you may not specify this mode along with enhanced remarks.

James P. Richmond
Medford, Massachusetts

Happy 25th Birthday, Commodore!

The Commodore Quarter Century: From Retail Shop to Global Giant

by Leslie Wood

In just 25 years a small typewriter sales and repair shop tucked away in downtown Toronto, Canada, has been transformed into one of the hottest personal computer companies in the world —Commodore International Limited.

Shipping more units world-wide than any other computer company, Commodore has grown from sales of $46 million in 1977 to over $680 million in fiscal 1983 (year ended June 30). And much of that success is due to the entrepreneurial instincts of Commodore's founder and present vice-chairman, Jack Tramiel.

The Polish-born Tramiel survived Nazi concentration camps to immigrate to North America and, in 1958, open his own typewriter shop in Toronto. Tramiel has always had a gift for anticipating future home and business electronic needs—and the ability to move quickly to fill them. Commodore's progress is a testimonial to that trait.

Over the past quarter-century Tramiel has led Commodore on a heady ride through adding machines, electronic calculators, digital watches and the introduction of the personal computer age. Together with his skilled management team around the world, he is still considering: what's next? Commodore, in fact, is widely acknowledged as a company that puts into action a smart but simple rule—hold onto the old for as long as it is good and change to the new the moment it becomes better.

During those early years, Commodore grew from typewriter repairs and sales to typewriter manufacturing, with the acquisition of a factory in Berlin, West Germany. Early in the 1960's Tramiel began selling and servicing a wide range of office equipment, and distributing nationally for an office furniture company.

In 1965 Commodore acquired the furniture manufacturer, and moved operations to what is now Commodore's present Canadian headquarters. Commodore still manufactures office furniture (mainly filing cabinets and desks, plus metal housings

for the CBM 8032 and SuperPET) at this plant in Scarborough, Ontario, and has expanded operations to three offices and two manufacturing plants in the Toronto vicinity.

Also in 1965 Tramiel met Canadian lawyer and financier Irving Gould, who later became Commodore's chairman. These two formed the head of the team that built the Commodore we know today. One of the first things this team did was to sell Commodore's adding machine plant and find a company in Japan to make adding machines for Commodore to distribute. While in Japan, Tramiel got his first look at an electronic calculator, and he quickly deduced that this product would mean the death of the mechanical adding machine. With the Commodore philosophy that "if we are not our own competition, then someone else will be", Tramiel moved quickly and found manufacturers to produce electronic calculators under the Commodore name. Thus the company was right there in the market when it began to take off.

The company began manufacturing its own electronic calculators in 1969 using Texas Instruments chips. In fact, Commodore was the first company to bring out a hand-held calculator—the C108—an example of what has become a long history of Commodore "industry firsts" in marketing value, innovation and performance in new products. It is interesting to note that this product was sold at much the same price, through similar distribution

channels and to similar customers, as is the popular VIC 20 computer today.

Up to 1974 Commodore expanded its line of calculators from simple four-function machines to memory machines, scientific machines and keyboard programmable models. At that time Commodore was largely dependent on third parties for the chips and displays that went into the products it was making.

Then in 1975 Texas Instruments decided to go into business against its own customers by manufacturing calculators. At the same time, chip prices dropped from $12 to $1 and Commodore was caught with a big inventory of chips and calculators while market prices plunged. It was this incident that led to Tramiel's decision that Commodore would be a company that controlled its own destiny, and not be at the mercy of other manufacturers.

Shortly thereafter, in 1976, Commodore purchased MOS Technology, Inc., one of its semiconductor chip suppliers, and worked its way toward becoming vertically integrated. This vertical integration now allows Commodore to supply its own needs and gives the company significant lead time in new product development. This means manufacturing cost advantages, which in turn translates into price/performance benefits for customers.

The acquisition of MOS Technology was followed in the next 18 months by two further key investments: the purchase of Frontier, a Los Angeles manufacturer producing chips that were complementary to those produced by MOS, and the acquisition of Dallas-based Micro Display Systems Inc., a manufacturer of liquid crystal displays. As a result of these acquisitions Commodore had in-house expertise and production in more key technologies than most electronics companies several times its size.

Also in 1976, Commodore reorganized its corporate structure as Commodore International Ltd. and moved its financial headquarters to the Bahamas and its operations headquarters to Wayne, Pennsylvania (it has since re-established in West Chester, Pennsylvania).

The next year was the watershed for Commodore when in 1977—still anticipating the future in true Commodore style—the company introduced its first personal computer: the PET.

The PET (Personal Electronic Transactor) uses the MOS-designed 6502 microprocessor, which is also used by some of Commodore's competition. It was this original machine, launched at the Hanover Fair in Germany and the Consumer Electronics Show in the U.S.A., that helped give birth to the personal computer market of today.

The PET sparked another period of rapid growth, which is still underway today. It was marketed worldwide and really took hold in the European market because of the widespread, loyal dealer network Commodore had developed in its distribution of calculators. Commodore dominates the personal computer market in Europe today with more than 50 percent of the market in many countries. In fiscal 1983 (year ended June 30) European sales reached $155.6 million (U.S. dollars), almost 23 percent of Commodore's total sales.

After the PET 4000 and later the CBM 8000 series micros, the next major product from Commodore was the very popular VIC 20. The prototype of the VIC 20 was previewed at the National Computer Convention in Chicago in 1980, and it was first launched in the Seibu Department Store in Tokyo, Japan because, as Jack Tramiel said about the threat of competition from Japan, "the Japanese are coming, therefore we must become the Japanese."

Commodore sold 800,000 VIC 20s world-wide in 1982, reached the one million mark early in 1983, and are now shipping VICs at the rate of 100,000 units per month.

Commodore didn't stop with that success either, but continued research and development and in August, 1982, shipped the first Commodore 64. By the end of that year, aided by the single biggest advertising campaign in Commodore's history, the 64 had already passed the Apple II in monthly unit sales. And by March, 1983, the 64 was being shipped at the rate of 25,000 machines a month.

Both the VIC 20 and the Commodore 64 are sold through mass merchandise retail outlets as well as computer dealers and selected electronics stores—a successful marketing technique that has since been

emulated by other companies.

Commodore has now become the largest unit seller of microcomputers in the world. And, according to a Dataquest study published in *Electronic News* recently, Commodore is number one in computers priced under $1,000 with an estimated 43% dollar share in the U.S. Maybe this is one reason why the *Commodore 64 Programmer's Reference Guide* is currently the top-selling computer book in the U.S.

In addition to the obvious success the company has achieved in the home market, the Commodore name is familiar in both the business and education markets for personal computers. Commodore is one of the leaders in small business computers with its SuperPET and CBM lines, and the 64 is also being used for a number of functions in small business.

The education market is another area in which Commodore is a front runner. In Canada, for instance, Commodore holds about 65 percent of the national market for computers in education. Penetration is also significant in U.S., British and European schools and universities.

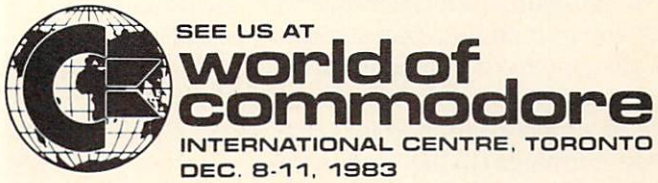
Commodore has become an international company, with manufacturing facilities in Japan, Hong Kong, West Germany, the U.K., Pennsylvania and California in the United States and Scarborough, a city within metropolitan Toronto. In fiscal 1983 world-wide sales increased 44.7 percent over 1982's $304.5 million to reach over $680 million. By the end of fiscal 1984, Commodore will be a billion-dollar-plus company.

Wall Street financial analysts who follow Commodore (shares have been traded on the New York Stock Exchange for three years, and were available on the American Exchange several years prior to that) state that much of the company's success is due to its flexibility and willingness to adapt quickly to—and even initiate—changes in technology and in the marketplace. Jack Tramiel puts it more simply: "The minute you're through changing, you're through."

Commodore is celebrating its 25th year with an international extravaganza being held in Toronto early in December. The "World of Commodore" show is the first truly international computer show to be orchestrated by a single microcomputer company.

The first all-Commodore show to be held in North America will feature 65,000 square feet of exhibits by suppliers of Commodore computers, software, peripherals and accessories, and by Commodore users clubs, special interest groups and microcomputer and business publications. Exhibitors are coming from several countries, including Canada, Turkey, the United Kingdom, Sweden, France and the U.S.A. to participate. Commodore operations from around the world will also be represented. In addition, a series of seminars conducted by some of Canada's best-known experts in the field will take the mystery out of micros for novices, and give valuable information to more experienced users.

A 10,000 square-foot hall will hold a major exhibit outlining Commodore's 25 years of history, its present hardware and software and the future of the company and its products. All who attend will see that the next 25 years will be as exciting as were the first 25. In fact, looking at the history of Commodore at the close of its first quarter century, it is easy to see that the company has consistently been a leader in recognizing change and leading the electronics industry into the changes. But, more than studying history, Commodore is a company that creates the history. Just watch. C



Easy Script 64, Commodore's Under-$50 Word Processor, Now Available at Your Dealer.

Commodore Software has announced the immediate availability of *Easy Script 64*, for retail sale. *Easy Script 64* is a full-featured word processor for the Commodore 64 and SX-64 (portable) color computers.

"This is the first time a word processor with this much power has been offered at such an affordable price," said Sig Hartmann, president of Commodore Software. "We want everyone to be able to afford a computer system and the most important software that goes with it—by important software, we include word processing at the top of the list."

Hartmann said *Easy Script 64* rivals the best word processing programs in home computing and includes features found only in business and professional systems, such as the ability to interface with a spelling checker (Commodore's *Easy Spell 64),* and transfer words, phrases and blocks from one section of text to another.
Features include:

—Change display colors
—Global/local "hunt and find"
—Global/local "search and replace"
—Goto line number
—Insert/delete characters, lines, blocks, sentences
—Optional sound effect prompts
—Print up to 240 characters per line
—Special function key editing
—Superscripts and subscripts
—Transfer words and phrases
—Vertical tabs as well as horizontal tabs
—View/Scroll 764 lines and 130 columns

In addition, a special "form letter" command lets the user create "personalized" salutations and body copy from a separate file by simply storing the information—usually names and addresses—in the file. A simple command tells *Easy Script 64* to insert the information in the form letter.

Easy Script 64 users can also add a spelling dictionary containing up to 30,000 words called *Easy Spell 64.* This companion software product points out possible misspellings by highlighting questionable words. In addition to the spelling checker's built-in 20,000 word vocabulary, *Easy Spell 64* lets the user enter up to 10,000 additional words — such as technical jargon or words the user commonly misspells.

Commodore Donates Computer Systems to Education Departments in Four States

As part of its CREWS (Commodore Resources in Education With States) grant program, Commodore recently donated a total of 120 computer systems to the State Departments of Education in four states: California, New York, Pennsylvania and Texas. The systems include computers, data storage units, printers, modems, and educational software.

The computer systems will be distributed by the State Departments of Education to educational support centers where they will be used for in-service teacher training and for evaluation of instructional software. The donated units will allow states to provide teachers with hands-on training.

Commodore dealers in the area of each training center have agreed to provide support for the donated units, and training for the program coordinators. Coordinators will, in turn, instruct the states' teachers.

Commenting on the importance of manufacturer and dealer support for education, William F. Kernahan, Executive Vice President Educational Systems (a Commodore dealer) writes in *Commodore* magazine (May, 1983), "The very real danger is that the microcomputer, a device that holds unlimited promise for education, could easily slide into either misuse or non-use simply because of lack of adequate support for the person on the firing line —the teacher."

The New York State Education Department's Commissioner of Education, Gordon M. Ambach, stated, "It (Commodore's grant program) will enable public and private agencies to co-operate in making the wisest use of available resources to resolve the overwhelming needs for teacher in-service training, while keeping in mind the complex dimensions of equitable access and delivery of training."

"Commodore through its grant programs is helping to give educators the up-to-date, technological training and support they need," said David Rosenwald, Commodore's Director of Education Sales. "It's part of an all encompassing effort by Commodore to further enhance its position in

schools. This effort includes a dedicated educational sales force, grants to State Departments of Education, and increased availability of quality software."

In addition to the CREWS grant program Commodore also offers a Matching Grant Program for Education. For more details on these and other Commodore support programs for education, write to Pat Walkington, Education Department, Commodore, 1200 Wilson Drive, West Chester, Pennsylvania 19380.

Dr. Dan Kunz Named Director of Educational Software

Dr. Daniel W. Kunz, 36, formerly Program Manager for Commodore Educational Systems has been named to the position of Director of Educational Software for the company. The announcement was made by Sigmund H. Hartmann, President of the Software Division.

Mr. Hartmann stated, "Dan brings the needed mix of educational, managerial and marketing skills to the newly formed Software Division. We are pleased to have him head up this important activity."

Dr. Kunz will be responsible for software development and acquisition of home and school educational software and related products. An educator by training, Dr. Kunz said, "The key word in the educational marketplace will be *quality.* Parents and teachers are demanding high quality software that uses the full capability of the computer to enhance learning at home and in school. Commodore intends to meet this demand."

Dr. Kunz has been a public school teacher, a college professor and a consultant to major educational foundations and agencies. In addition, he was a National Policy Fellow in Education at the United States Department of Education. Prior to his employment at Commodore he served as Director of Curriculum Planning and Inservice for the New Jersey State Department of Education where he also coordinated activities of the Teacher Corps, Teacher Center and National Council of States on Inservice Education. C

Spread Sheet Programs: Expensive Gimmicks or Management Tools?

by Donald E. Hassler
Fidelity Management Systems Phoenix, Arizona

This is the third in a series of articles about Commodore microcomputers and business. These articles are designed to tell business owners just like me how to get the most out of that devilish device you purchased to make your daily business activities more productive.

I have been using a Commodore 8032 for almost three years. During that time I have tried just about every type of program there is. I have also written some programs of my own. And I have modified others' programs.

There aren't many short cuts to getting the most out of your computer. You must get used to it yourself. The microcomputer is a *personal* device. So that is lesson number one: sit down in front of it, slide in a disk, and *get going!*

Now, what about spread sheets? I have been using *VisiCalc* for many months. It is simple, fast and powerful. There are assuredly more things you can do with it than you can possibly imagine. So load up your own favorite spread sheet and let's learn a few tricks.

If your business depends on sales or production of any kind, that's your first program. Lay out a complete sheet showing employee names, quotas or objectives, actual performance and percentages. Do it for each month and for year-to-date(YTD). This may seem a simple start but it works. It also gives you a regular report that your clerk or bookkeeper can do for you. But it's easy, gets you used to the system and offers regular information to your employees. (Lesson number two is that most of the initiative for new reports that you want to try will have to come from you.)

Now let's get more complicated. One of the most important uses for spread sheets is budgeting and forecasting. You are probably doing it already, but on paper. Well, throw away the pencils and erasers. Here's the easy way. Once you get all your accounts or headings on the sheet you never have to do it again. And if you're not sure about the exact setup, buy one of the excellent books which show sample *VisiCalc* (or other) overlays.

Start with the sales budget and the expense forecast. Put all your major headings on the sheet and then fill in the columns with numbers. You can do it for the month and for YTD and have all that come out on standard 8½-inch wide paper. Or if you want a really big sheet, run 132 columns and condense your printer output.

Here are a few tips for setup:

- You'll find that numbers are easier to read if rounded off to even dollars, so use the "/FI" function (integer format)
- Remember that the columns can be varied in width, so experiment with proper widths to get everything you want on one sheet.
- Don't forget to show percentages or dollars over (under) budget. All this figuring will be done for you by the program.
- For speed of entry, set up your program so that it doesn't recalculate the whole sheet every time you add a character or a number (manual recalc). That's "/GRM" in *VisiCalc*.

Now you're ready to try something a little more complicated, but it's worth it. This routine, a "break even" analysis, will reveal more about your business than almost anything else. It also makes the best use of spread sheet systems because you can vary all the figures on the sheet with only one master entry.

Start by setting up a column of labels for all your sales/revenue functions, cost of goods functions and a complete list of all your expense categories. Remember to add expenses in two types, fixed and variable. Fixed are those which are always the same: rent, various lease payments, insurance, etc. Variable are those that change with some other function: sales commissions, advertising, p/r overhead, etc.

Now fill in the column of expense items with dollar values for the fixed items and a percentage formula for the variable ones. The formula will be a percentage of either sales or costs (or maybe even

payroll in the case of p/r overhead). Now find the position in your column for total sales/revenue. That number is the key to your analysis. Everytime you change it, all the variable items will change by their proportionate percentage. When sales rise, commissions, advertising and promo expense all go up, for instance. Fixed items like rent stay the same whatever your revenue (unless you're on a percentage rent, in which case figure it in.)

You want to find a spot near the top of your sheet and have the total net sales figure copied there. Now you can change it easily and watch how it effects the bottom line (net profit). After locating the break even sales figure you can play with the sheet even more. Try varying the cost of sales items. Or try changes in your gross margin percentages.

Total familiarity with this approach is a must for any business manager today who wants the most out of his Commodore computer system. So this is the biggest lesson for this article: find ways related to your business where you can use the power of your Commodore. Changing long and complicated rows and columns of figures instantly is one of the most effective ways to use microcomputer power.

See your local Commodore dealer for recommendations on the best spread sheet program for you. C

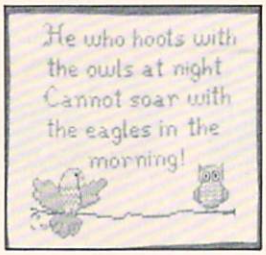
Lincoln College Commodore Computer Camp

by Tom Zurkammer

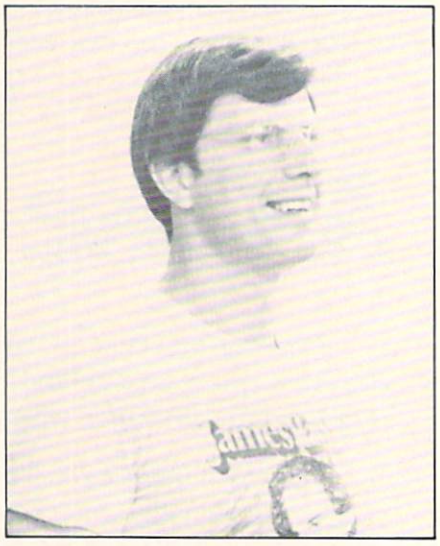
Keith Peterson instructs his students in advanced BASIC and assembly language.

Fifteen-year-old Jon Sadler, one of the camp's youngest participants, came to the camp to learn assembly language.

Jim Butterfield reflects on the answer to a question in one of his sessions covering the Commodore 64.



Computer hackers, take warning!



Dick Immers, the "Disk Doctor", teaching file handling and advanced disk operation.

Commodore users found more in the Central Illinois prairie than corn and soybeans this summer. They also discovered a small slice of computer heaven thanks to Lincoln College's first Commodore Computer Camp.

Lincoln College, a small, private, two-year college located in Lincoln, Illinois, hosted some 50 "campers" as well as Commodore computer experts for a week-long workshop last summer. Participants traveled great distances—some from as far away as both coasts, Texas and Canada—to rub elbows, swap notes, attend classes and lectures and bask in the knowledge of such noted Commodore "wizards" as Jim Butterfield, Dick Immers, Walt Sadler, Keith Peterson, and Len Lindsay.

The adult campers displayed varying degrees of computer knowledge. Most participants, however, were highly computer literate. They use their computers in small businesses, in corporations, in teaching and as hobbyists.

Jim Strasma, assistant professor of computer science at the College and editor of the *Midnite/Paper,* and I conceived the idea of the camp. We wanted to bring together several well-known Commodore computer experts at one time in one place and introduce persons from across the country to Central Illinois and Lincoln College, because we have a strong commitment to building a top-notch computer program at our college.

Income derived from the workshop will help expand the college's existing Commodore equipment and, more importantly, provide funds for student scholarships. The college's three year-old computer department recently bought 16 SuperPETs to enable the offering of different computer languages to about 150 students per semester on the main campus. Other classes are conducted at the college's branch locations.

The presence of Jim Butterfield, whom Strasma calls "the most famous PET owner," proved to be the trigger needed to draw attendance at the camp. A well-known technical writer from Toronto, Ontario, Butterfield's appeal is in his ability to take something complex and make it understandable.

"They tell us he's the man who keeps Commodore honest. He's not on their payroll and he tells the truth, good or bad," commented camp participant Betty Clay, a math and science teacher in the Arlington (Texas) Public Schools. "I came to this camp because of Butterfield. He knows how to teach. I expected to understand him and I did."

Butterfield's classes focusing on assembly language and sound and graphics on the Commodore 64 were among the best attended camp sessions. Participants eagerly absorbed information about assembly language, primarily to be able to write better game programs using machine language instead of BASIC. Butterfield's sound and graphics sessions for beginners taught control of those functions using BASIC.

Butterfield quickly refutes the conception that computer people isolate themselves from the rest of society in an effort to keep their knowledge to themselves.

"The idea of a mad scientist sitting in a corner with a computer does not in any way resemble reality. Computer people are very social. They like to talk about their computers, brag and share their knowledge," he said. These feelings might be well evidenced by the "PUGs" around the country (Pet User Groups).

Dick Immers, known in Commodore circles as the "Disk Doctor" because of his doctoral research on computer disks, was another of the featured Commodore experts. Disk file handling and advanced disk operation sessions taught by Immers also proved popular. He helped participants refine their file handling skills, which are essential for successful programming. The advanced sessions explored the disk drive using machine language.

Walt Sadler, a professor of mathematics at the University of Wisconsin, Waukesha, who specializes in teaching computer use to young children, conducted sessions for beginners and persons interested in knowing more about computer applications for schools.

Keith Peterson, a former member of Commodore's software development and support group, conducted sessions in advanced BASIC and advanced assembly languages. Peterson used an assembler to demonstrate machine language programming. He also presented information about Commodore's "B" machine and CP/M.

Peterson noted that Commodore supports opportunities for users to share knowledge. "The millions of people using Commodore will have a lot more knowledge than the computers' designers," he said.

An introduction to COMAL was demonstrated by Len Lindsay in an evening presentation. COMAL is a computer language that combines the best parts of Pascal and BASIC. Lindsay had with him one of the first copies of COMAL for the 64. It promises to be a fine language, especially for schools.

Debra Landre, a computer instructor at Lincoln College's Bloomington campus, taught intermediate BASIC sessions. She brought participants, who already had at least one BASIC course, to a more skillful level of programming. She also provided help with

education

Interestingly, a real sense of how to assemble programs with more style.
community surfaced at the Lincoln College camp.

"We are a group working together for our mutual benefit. You really don't ever become an 'expert,' because you are constantly learning new concepts and expanding your knowledge outward," Peterson remarked. "You know you don't know everything, but you know how to figure things out."

One of the youngest camp participants, 15 year-old Jon Sadler, son of Walt Sadler, came to Lincoln College's camp to learn assembly language. The young Sadler began working with computers when he was nine years old. Now he has several programming awards under his belt. He recently co-authored, with his father, a soon-to-be marketed program for teaching music.

"I consider the computer a challenge. I like to see if I can do it. Often I don't get my homework done," he admitted.

Jon Sadler enjoys the label "computer nut." To be a computer nut, Jon says, "you need to know a computer language and be devoted to getting the program right or working to debug it and try again, but not abandon the program. You also have to be in the abstract stage of mind development in most cases."

J. Paton Dellow, Lincoln, Illinois, who reviews computer software for Commodore and works as a supervisor of clinical services at a Central Illinois correctional facility, felt the camp was nothing short of "fabulous."

"There is a great deal of talent at the camp. I hope to capitalize on as much of it as possible," he said.

What surprised the camp organizers was the level of emotional and financial investment exhibited by camp participants.

"It turned out to be a very high plane," Strasma noted. "As one person commented, 'I came to eat, sleep, drink and breathe computers,' and that's what the majority of participants appeared to be doing. Even scheduled free time ended up being devoted to computers at the participants' choice. The camp won rave reviews."

Lincoln College staff are already planning next year's summer computer camp which will again feature Commodore experts. Enrollment will be limited because we want to start out small and be very personal. A complete listing of course offerings and times will be available in late November. Information about next year's camp can be obtained by writing to Tom Zurkammer, Dean of Academic Affairs, Lincoln College, Lincoln, IL 62656. C

The Commuting Commodore

by Doris Dickenson

Our classroom 64 gets a "baby brother"

Our fourth-grade classroom treasury got a real boost from an aluminum can drive last spring and by watching for the microcomputer sales we were able to purchase a VIC 20 as a second classroom computer. I made this choice because the hardware we already had and the programming the students had already learned on the 64 would be compatible with the new computer.

We set up a second computer corner in the back of the classroom and copied the manuals we were already using with the 64. Except for the color codes, our material applied just as well to the VIC 20. I checked the practice programs the students were using and coded the upper corner of each page 64 or VIC 20 or both, so students could tell readily which programs were appropriate for which computer.

I really fell in love with the VIC 20 myself. When I taught programming, with my monitor on display for the entire class to watch, the large type and clarity of image made my examples so much more visible!

But I still wasn't satisfied. I wanted to give the students even more of a personal tie to the new computer, because it would be staying in the class when they moved on at the end of the year and they had earned the money to purchase it. That's how the "commuting computer" idea came about.

We had our new computer, and a letter of explanation, available for viewing at our open house in the spring. It told the main requirement for bringing the computer home overnight or on weekends: signing the Parental Computer Agreement. A sample of this agreement follows this article.

The names of eligible students, those whose parents had signed the agreement, were drawn at random each Friday, which established the computer schedule for the following week. Students who had had a turn waited until all eligible students were chosen before they got a second turn. Any switching of nights between students selected for that week had to be done among the parents. I did not want to be involved in it. This worked very satisfactorily when any family had a conflict with dates.

Using the original packing box as a carrying case, I included not only the regular user manual, but a simplified diagram of the hookup method, in case

PARENTAL COMPUTER AGREEMENT

Room 12, Pine Crest School
Sebastopol, California

I agree to be responsible for the Room 12 computer. I understand the following conditions for borrowing it:

1. No food or drink is to be used around it because it can be damaged permanently by anything in the keys.
2. The computer will be picked up at school and returned to school by private vehicle. It is not to be carried home or taken on a school bus.
3. The computer must be returned as scheduled on the following day, or after a weekend, to give the next student his turn.
4. Small children will use it *only* under adult supervision.
5. If the computer is returned in damaged condition, I agree to replace it.
6. Violation of conditions 2 or 3 will result in your student losing computer privileges for the remainder of the year.

Student_____

Parent Signature_____

Date_____

some family did not take time to read the manual carefully. I also included a copy of the simplified manual the students used at school along with some of the programs with which they were already familiar. That really made them stars at home. Because of a recent article I had read warning of the dangers of mixing polarity in some wiring, I included a triple plug and encouraged them to use it for the computer and monitor, just to reduce the chance of a problem.

Of course, our "commuting computer" was a huge success. It was the talk of the school among parents as well as students. It was great P.R. for our entire computer program. And the computer seemed none the worse for its adventures. This year we plan to continue the same program. One thing will change, however. The commuting program will start much earlier in the year, and it will be part of the reward system as students move through the sequential programming lessons that we use. C

Educational Programming: A Method

by M. W. Caprio

If you're a teacher who writes programs, here are some helpful hints to speed things up.

In a way, the advent of affordable personal computers on the education scene has created a new problem for classroom teachers. In fact the computer, purported to be a time saver, is often —quite to the contrary—a great consumer of teacher time. The problem revolves around software and is a kind of "software dilemma."

With good programs a personal computer can be a powerful educational aid. Getting good programs is the crux of the problem. If the programs are to do what teachers want, they really ought to be written by teachers. But teachers are already busy teaching. Writing high quality programs takes time —it takes time away from the study, lesson planning, extracurricular activities, all the things that teachers already do. But teachers are trying to write programs, nevertheless.

Working under the time constraints of a demanding profession, they are often forced to omit the subtleties that would be desirable if included in their programs. A title page, sophisticated reinforcement, and student response analysis can make the difference between a valued professional tool and an amateur attempt at one.

Professional programmers include all of these niceties, and more, in the commercially available products that they produce, but professional programmers are not teachers. Some may have once taught, but now the program —not necessarily the student— is their chief priority. A subtle difference, perhaps, but one that impacts heavily on the teaching value of commercially available software. These programs may sparkle but, too often, they force us (teachers) to a pedagogical compromise.

And so it happens that educators are thrust on the horns of the "Software Dilemma". Will it be hastily written programs with that unwanted homemade touch, written with stolen time and reflecting the time constraints that hurried their composition? Or will it be a program with all the bells and whistles that the professional can provide but without an acceptable educational philosophy or effective teaching style? Neither is acceptable. Amateurish programs are inconsistent with professionalism in education, and pedagogically unsound programs may actually do more harm than good. Fortunately there is a solution.

To get good educational software we must do one of two things: either teach programmers to teach, or teach teachers to program. The latter is obviously the more realistic choice. As teachers, we must learn to write professional-quality programs that are commensurate with our professional status as educators. And we must learn to write quickly and efficiently so that our other responsibilities will not suffer disproportionately to the benefits to be gained by programming. After all, it has to be worth the time we spend on it. The objective then is to get to the point where programming is time-efficient, where we rely less on professional programmers and where we improve the quality of our programs. There is a programming method that makes it possible for anyone with even a rudimentary knowledge of BASIC to achieve this goal.

Teachers can most quickly enhance the efficiency and quality of their programming by adopting the logical approach to programming that many professionals use. Begin by writing your programs in a modular format. A modular program is a collection of subroutines that are called by the main part of the program to perform the tasks that need doing. In essence, writing a program in subroutines is a way of breaking a big job down into its smaller, more manageable components. Teachers are especially good at seeing what those components must be because in much of our teaching we do exactly that. We teach subject matter to our students in manageable "subroutines".

But, you say, how can this improve efficiency? Wouldn't the modular program need to be as long or longer than the linear version of the same program? Valid

questions. What makes this work for programming efficiency is something I haven't said yet. That is, *keep records.* Every time you develop a new subroutine, or find one you can use in a magazine or borrow one from another program, you must write it down. (I keep mine in a loose-leaf binder so they can be kept in alphabetical order as new ones are added.) By the time you've developed two or three programs you will have quite a collection of subroutines.

These subroutines might include your title page, reinforcement for student answers, sound, computation of the student's score and many others. They are all routines that can be used in more than just one program. What is most important is that they will all reflect your own teaching style and will correlate nicely with the way you do things in the classroom. They will use your language, your humor and will fall safely under the umbrella of your educational philosophy. They will precisely reinforce what is being done in your classroom: no more, no less.

Before long, writing a new program will mean writing the main program (it will bristle with GOSUB statements) and modifying the subroutines to fit the task at hand. You will be able to make most modifications as you type them in from your notebook. It will be unnecessary to reinvent the wheel each time. Drawing on your subroutine library reduces duplication of effort to near zero.

Shortly after beginning this method I found that the quality of my programs began to increase dramatically while, at the same time, my writing got faster. The quality improves, I think, because after becoming familiar with a subroutine by using it a few times, I begin to see ways of refining it that may not have been obvious to me the first time through. Then too, student suggestions have been incorporated from time to time. Improvements continually accumulate in my loose-leaf book. Over the years I've polished and repolished my subroutines until they've become real gems. This is, clearly, not the tack to take if you are only planning to write one or two programs. Its rewards are not immediate but accrue over a semester or two.

Like the ditto machine, overhead projector and pocket calculator, the personal computer has become part of the teaching technology—a tool of the trade. Whether or not it will be integrated into our professional lives is no longer an issue. The real question now is how to best streamline its assimilation.

To get you started with modular programming I would be happy to send you copies of a few of my treasured subroutines for the PET and Commodore 64. They are simple but effective. Write to me at Box 180, Stony Brook, New York, 11790 and send a self-addressed, stamped, business envelope.

While you are waiting for the mail to bring your subroutines, you may want to try the sample program. It took ninety minutes to write using routines from my

loose-leaf book and is included as an example of modular programming. The routines that you see are accessed by GOTO as well as GOSUB statements, whichever works better.

Each routine is identified by REM's but I'd still like to say a few words about two of them here. First, there is the Input Subroutine. It is used in place of the INPUT statement to prevent those accidental crashes so frustrating to new computer users—the ones that occur when they press the PET's return key at the wrong time. Since it gets the input in a string variable, it obviates the REDO FROM START error. Use it to help crash-proof your programs, or modify it to—in effect—disable any key(s) for input, with the exception of RUN/STOP.

The Character Search routine is the second. I use it to analyze student responses for the inclusion of a particular character. In this program it is looking for a dollar sign but it can be easily modified to search for commas, apostrophes or any character at all. In some of your programs you will want to alter it to check for a whole class of characters, like capital letters. Use the Character Search routine to elevate your program's interactive style and enhance its capability to provide feedback to students.

The sample program is useable and may even be useful as it stands. But, it wouldn't hurt to dress it up a bit with one or two subroutines of your own. A title page would be a nice embellishment and a page of instructions may help to make the program a bit more user friendly—two easy subroutines. Other alterations might include modifying the subject matter itself.

How about making it into a drill for percent problems or an exercise for students to practice their number facts? It is a parent program that can easily give rise to a whole family of offspring. For example, I can modify the sample to do, let's say, addition facts in much less time than it took me to write the original version. Then, the hardest part—but not the only part—to create *Subtraction Facts* from *Addition Facts* will be changing the title page. The program family grows so easily and so quickly because the "new" programs are built of previously developed routines. That is, after all, the whole idea. **C**

Subroutines

```
380 REM *** INPUT SUBROUTINE ***
390 Z$=""
400 GET A$:IF A$=""OR A$>CHR$(90) OR (A$=CHR$(20) AND LEN
    (Z$)<1) THEN 400
410 IF A$=CHR$(13) THEN 460
420 IF A$=CHR$(20) THEN Z$=LEFT$(Z$,LEN(Z$)-1):GOTO 450
430 IF A$<CHR$(32) THEN 400
440 Z$=Z$+A$
450 PRINT A$;:GOTO 400
460 RETURN
470 :
480 REM *** CHARACTER SEARCH ***
490 FOR I=1 TO LEN(R$)
500 B$=MID$(R$,I,1)
510 IF B$="$" THEN DS=1:GOTO 530
520 T$=T$+B$
530 NEXT I
```

```
540 R=VAL(T$):T$=""
550 RETURN
560 :
570 REM *** INST. ON ERRORS ***
580 REM THIS WILL BE MORE SOPHISTICATED FOR MORE COMPLEX
    EXERCISES.
590 PRINT"[DOWN]$";M1;"-[SPACE]$";M2;"=[SPACE]$";A
600 PRINT"[DOWN]YOU[SPACE]HAVE[SPACE]$";A;"LEFT."
610 RETURN
620 :
630 REM *** ENDING ROUTINE ***
640 PRINT"[CLEAR]YOU[SPACE]TRIED";N;"PROBLEM";
650 IF N>1 THEN PRINT"S";
660 PRINT",[SPACE]"+N$+"."
670 PRINT"YOU[SPACE]GOT";C;"CORRECT."
680 PRINT"[DOWN]THAT'S";INT(C/N*1000)/10;"[LEFT]%"
690 PRINT"[DOWN2]BYE."
700 END
```

Sample Program2

```
100 REM *** A SAMPLE PROGRAM ***
110 :
120 PRINT"[CLEAR]WHAT[SPACE]IS[SPACE]YOUR[SPACE]FIRST
    [SPACE]NAME?[SPACE]";
130 GOSUB 390:IF Z$=""THEN 130
140 N$=Z$
150 :
160 REM ***  THE MAIN PROGRAM  ***
170 N=N+1:M1=INT(20*RND(1)+11):M2=INT(9*RND(1)+1):A=M1-M2
    :DS=0
180 PRINT"[CLEAR]";N;"[LEFT]."TAB(5)"IF[SPACE]YOU[SPACE]
    HAVE[SPACE]$";M1;"AND"
190 PRINT TAB(5)"YOU[SPACE]SPEND[SPACE]$";M2;"OF[SPACE]
    THEM,"
200 PRINT TAB(5)"HOW[SPACE]MUCH[SPACE]MONEY[SPACE]IS
    [SPACE]LEFT?[SPACE]";
210 GOSUB 390:IF Z$=""THEN 210
220 R$=Z$:PRINT
230 GOSUB 490
240 IF R=A THEN 300
250 :
```

```
260 REM *** REINFORCEMENT ROUTINE ***
270 REM  A MORE SOPHISTICATED ONE IS AVAILABLE FROM THE A
    UTHOR.
280 PRINT"NO.":GOSUB 590
290 GOTO 320
300 C=C+1:PRINT"[DOWN,RVS]VERY[SPACE]GOOD."
310 IF DS=0 THEN PRINT"[DOWN]...BUT[SPACE]YOU[SPACE]
    FORGOT[SPACE]THE[SPACE]DOLLAR[SPACE]SIGN."
320 PRINT"[DOWN3,SPACE5]PRESS[SPACE,RVS]C[RVOFF]ONTINUE
    [SPACE]OR[SPACE,RVS]E[RVOFF]ND."
330 GET C$:IF C$=""THEN 330
340 IF C$<>"E" AND C$<>"C" THEN 330
350 IF C$="C" THEN 170
360 GOTO 640
370 REM:
```

Commodore's subsidiary, MOS Technology, Inc., produces the chips that are the heart (and other such indispensable parts) of your microcomputer. Here's how they do it.

IN THE

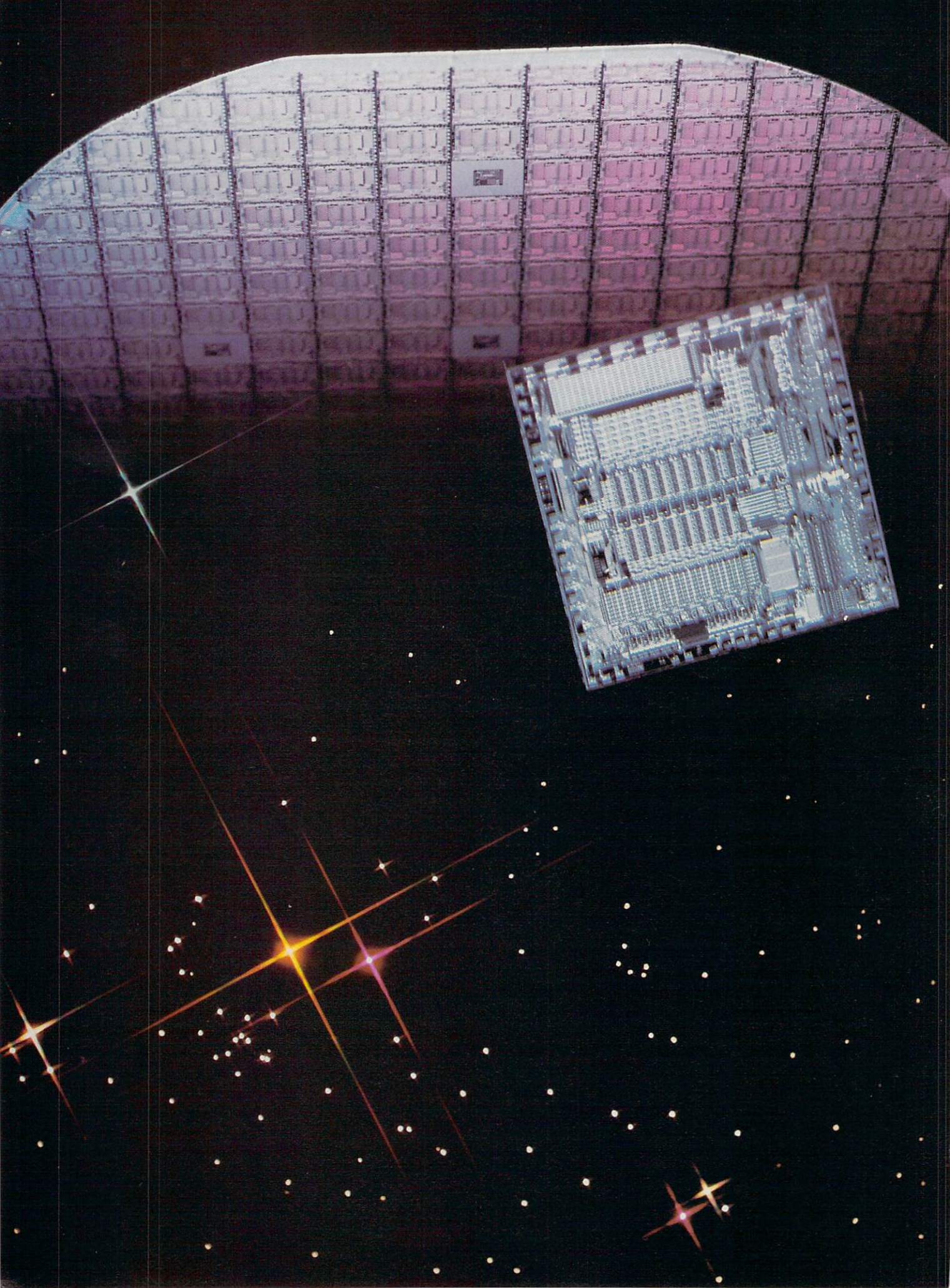
By Diane LeBold

If you've taken courses in integrated circuits most of this will be no news. But if you, like me, have been stumbling over words like "semiconductor" without understanding quite what they mean—or have been wondering exactly what this thing they call a "chip" really is—read on.

Those of you who have opened your computer's case know it's crawling inside with what looks like a bunch of mechanical caterpillars. Maybe somebody told you, as somebody told me, that these are your computer's "chips". Close, but not quite accurate. Although these caterpillars *contain* the chips, they themselves are just the protective packaging. They are called, in technical terms, "DIPs" (Dual In-Line Packages). When a chip is contained in a DIP (which sounds like something you'd find at a cocktail party rather than inside a computer), the whole thing is called a "device".

The parts you *don't see*—the quarter-inch silicon flakes inside

PHOTOGRAPHY—ROBERT EMMOTT



the DIPs—contain the actual electronic circuits that run your computer. There are several different kinds of circuits in your computer, each one with a different job. If a circuit carries out commands and does calculations, we call it a microprocessor. If it simply stores information it's a memory circuit. If it lights up the tiny dots on a monitor it's, of course, video. And if it controls speakers it's a circuit for sound.

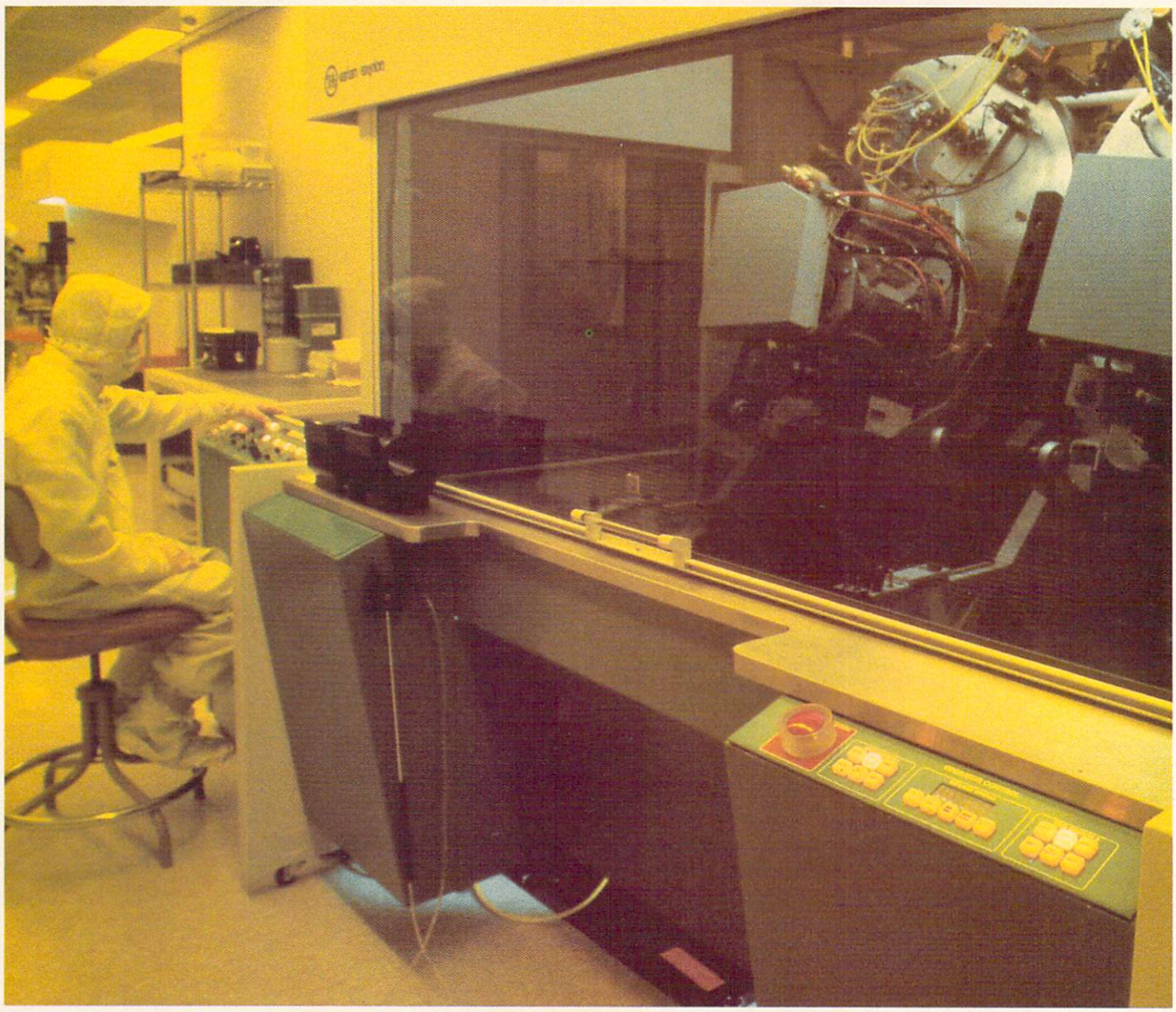
The miracle (taken for granted now) is that these complex circuits, which thirty-five years ago would have taken up literally whole rooms and used the power of hundreds of lighthouses, have been reduced to microscopic lines on a piece of silicon half the size of your little fingernail, using the power of a child's nightlight.

A Sort-of-Short History

Many of us (I hope) remember when televisions and radios were run on vacuum tubes. They were large (certainly far from portable) and gave off a lot of heat. Do you recall when the switchover began, first to transistors and then to "solid state"? I was only a kid, but I remember how progressive it was in the mid-fifties to have a transistor radio (look, Ma, no tubes!) that you could actually *carry around* with you. And then in the sixties came solid state televisions—also smaller, lighter—and significantly less power hungry—than the old ones.

Behind the scenes what had happened was the invention, first, of the transistor in 1947, and then finally of the integrated circuit (IC) around 1960—both using the properties of semiconductors to create, first, electrical components that replaced vacuum tubes and finally complete electrical circuits.

Let's stop here for a minute and talk about semiconductors and integrated circuits, since they are the crucial developments that made all this possible to begin with. A semiconductor, as its name suggests, is a material that is not quite an electrical conductor (like copper is) and not quite an insulator (like glass or rubber), but can sometimes be one or the other. I know that sounds like something out of Lewis Carroll, but it's not as wacky as it seems.



In the metallization process wafers receive a coating of silicon dioxide or other material.

High-Tech Glossary

Bi-polar transistor A transistor formed by sandwiching either a negatively doped region between two positively doped areas, or vice versa.

CAD (computer-aided design) Used as a tool in designing integrated circuits. Information about past circuits, stored in a data base, is combined with data about the function of new chips to enable the CAD system to design a new circuit.

Chip The nickname for an integrated circuit formed on a tiny piece of semiconductor material.

Diffusion furnace A high-temperature furnace used in making chips. It diffuses dopants into silicon wafers, which transforms the silicon into a semiconductor material.

DIP (dual in-line package) The protective packaging, resembling a caterpillar with stubby metal legs, that houses a chip. It is usually made of plastic but can also be ceramic.

Doping The process of selectively introducing impurities (called dopants) such as phosphorus, boron or arsenic into a pure material like silicon or germanium to create a semiconductor.

IC (integrated circuit) A group of inseparably connected circuit elements formed on and within a single substrate, usually silicon.

Ion Implantation A very precise method of introducing impurities into silicon to create a semiconductor material. Single ions of dopants are shot into the silicon under carefully controlled conditions.

LSI (large-scale integration) Generally applied to integrated circuits containing 40,000+ transistors.

Microprocessor An integrated circuit designed to carry out commands and do calculations.

MOSFET (metal oxide semiconductor field-effect transistor) A transistor formed by creating islands of negative and positive silicon connected by a channel of silicon dioxide, over which a layer of metal is deposited.

Photomask Used as part of the photographic process in making chips. It protects (masks) the areas that should not be exposed to light, so those areas can later be etched.

Planar process A commonly used method for creating integrated circuits. A layer of silicon dioxide is formed on the surface of a silicon wafer and is then photolithographically patterned to permit the introduction of dopants.

Plasma etching A process in chip making. After a wafer has gone through photolithography certain areas are etched using an excited chemical vapor called a plasma.

Reticle A picture in negative of one layer of a circuit. Reticles at MOS Technology, Inc. are created on glass coated with photosensitive emulsion and are usually ten times the final size of the circuit.

Semiconductor A material that is less of a conductor than copper but more of a conductor than glass, whose electrical properties can be altered to suit specific needs.

Silicon An element used in pure crystal form as the base material for many kinds of integrated circuits.

SINCAP The final protective coating on a wafer. Composed of silicon nitride, it prevents damage to the microscopic circuits.

Solid state Electrical functions carried out in a solid medium.

Transistor A semiconductor device that acts as either an amplifier or current switch.

VLSI (very large-scale integration) Integrated circuits that contain as many as 100,000 transistors and up.

Wafer A thin disk of semiconductor material, usually about five inches in diameter, on which hundreds of identical chips are fabricated.

Yield The percentage of usable chips produced by the wafer fabrication process.

Semiconductors are made (they are indeed made, not born) by adding specific impurities (usually boron, arsenic or phosphorus) to an extremely pure material like silicon or germanium. (The process of adding the impurities is called "doping".) The impurities generally cause the silicon (or germanium) to conduct electricity when it ordinarily wouldn't. If you place the impurities very precisely (you'll see what I mean by "precisely" a little later) you can make electricity do whatever you need it to do—stop, go, amplify—whatever.

The most effective semiconductors developed so far are metal oxide semiconductors, which is where the acronym MOS comes from. And the most effective transistors are metal oxide semiconductor field-effect transistors, also known as MOSFETs. Just in case you ever need to know.

Integrated circuits, which are complete electrical circuits made out of one piece of semiconductor material, grew up almost simultaneously with semiconductor technology. In an integrated circuit, all the components are fabricated on and within a single "substrate", which, in the case of your computer's circuits, is usually silicon.

Contrast the integrated circuit with other circuits you may have seen that are composed of "discrete" or separate components —resistors, capacitors, transistors (or vacuum tubes, if they're really old)—all wired together. You'll notice one thing right off—integrated circuits can be a whole lot smaller. That means, for one thing, that they're faster (less distance for the current to travel) and draw a lot less power. They're also a whole lot cheaper to produce. And, as fabrication processes continue to be refined, they keep getting even smaller—and cheaper.

I won't go beyond that, since we'll take a close look at the process a little later as we tour MOS Technology. But before I change the subject, I should mention that people were messing around with semiconductors for quite a while

before Bardeen, Brattain and Shockley at Bell Laboratories finally figured out, in 1947, how to make a successful transistor. (That discovery, coincidentally, was made almost exactly 36 years ago, between November 16 and December 17.) In fact, in the early 1900's—several years even before the invention of the vacuum tube—people had been trying to use crystal detectors (made from semiconductor materials) in radios, but their performance was too erratic to make them marketable.

The early transistors, as you may remember, also had their limitations. They looked rather like little three-legged stools, and were inclined to fall off the circuit boards. However, once transistor technology got rolling, discovery rapidly followed discovery until in the mid-1950's engineers learned to use photolithography to define transistors on the surface of a solid such as silicon (more on this later). Then they realized that all necessary electrical components—not just transistors, but also resistors and capacitors—could be made from semiconductor material and fabricated in place in a solid, all of which led finally to the creation of integrated circuits on (and in) solids like silicon—and the now familiar microchip.

As you might suspect, much of the original support for creating miniature electrical components came from the military establishment, which was looking for a way to put a lot of complex electronics into their weaponry and equipment. Obviously, existing electronics like the 30-ton ENIAC computer that was around in the 1940's wouldn't fit well in a bomber, so after World War II various branches of the military had begun supporting a fair amount of research toward scaling things down. Naturally, the

Wafers are carefully cleaned before each operation.

first attempts were at making miniature versions of conventional components, none of which was very successful.

Ironically, however, when the integrated circuit was finally developed in the late fifties, the Air Force was already supporting a different branch of research called molecular electronics and the Army was committed to work on a "micro module", which used miniature electron tubes in a ceramic substrate. Consequently, as the politics of these things go, the integrated circuit was mainly ignored for a while.

Nevertheless, the story of the IC, as we all know, had a happy ending in spite of early neglect. As fabrication processes were refined and chip "yields" increased—that is, more chips made it successfully through fabrication in working order—it became impossible to ignore the fact that IC's were cheap, doing for pennies what used to cost hundreds of dollars. So even though it took a while to catch on in any far reaching way IC's were used in military and space applications fairly early on. Then you began to find them controlling production processes in factories, and showing up in consumer appliances like televisions, washing machines, dishwashers, fire alarm systems and even toys.

Then came large scale integration (LSI), which meant more transistors and more circuitry on a chip. How much more? Well, in the early 1960's IC's contained maybe 400 transistors. On the other hand, LSI circuits contain maybe 60,000 transistors—a number that has consistently been climbing.

Now we get to the good part—1971. That's when Ted Hoff at Intel, which was then a two year-old company, developed the first IC that worked as a microprocessor. It could do more calculations faster, all in a quarter-inch flake, than our old 30-ton friend ENIAC could do only 25 years before. This is where we came in.

From Sand to Microcomputer: How a Chip is Made

Actually MOS Technology, Inc., the subsidiary of Commodore that produces the chips that go in your computers, made other semiconductor devices before they started making microcomputer circuits. Originally the chips they made were for controlling home appliances, factory production processes and other kinds of electronic systems. Then in 1974 they developed the 6502 microprocessor, which is really where we came in.

However, whether the IC on a chip is running a washing machine or a computer, the basic fabrication process is the same. The process is similar, in a general way, to silk screening or batik. In those processes artists expose certain parts of a design to color, while protecting other areas with wax or a similar substance. Then they switch, and expose new parts of the design to a different color while protecting yet other areas. They build up colors by protecting and exposing until they get the effect they want.

Similarly, layers of circuitry are built up on the surface of a chip using a photolithographic process that exposes certain parts of the silicon to specific chemicals but protects others, then switches and exposes *different* parts of the surface to perhaps *different* elements while protecting yet others—until all the layers are in place.

Design

The first step in designing an LSI circuit is to define the "architecture" or what the chip is supposed to do and how it will do it. The design engineer then uses computer simulations to help define the exact circuitry requirements and a layout specialist draws the final circuit—on a much larger scale then the final chip will be, of course. However, on a device that might include 100,000 transistors, you can't draw every element. So the layout specialist will draw, for instance, one cell of a 64,000-cell random access memory, which is then digitized, converted to a computer data base and duplicated with the help of a CAD (computer-aided design) system.

Once the circuit is edited and stored in a new data base, the computer checks to make sure no design rules have been violated. This checking process alone could take years if done by hand, but takes the computer only a few hours. A variety of cells, each representing a different function, can also be stored in the computer at this time for use in future circuits.

Finally the complete design data base is converted to magnetic tape and sent to the first step in the fabrication process—photomask.

Before we talk about the photomask process, however, you need to know a few things about wafers, since every chip starts out on a wafer—along with hundreds of sibling chips. Wafers are just what their name implies—thin, round slices of pure silicon. They're made by melting sand, removing the impurities and crystalizing the remaining silicon into a single-crystal log about five inches in diameter. The log-shaped crystal is then sliced into thin disks, polished to mirror smoothness and voila! a wafer. One wafer becomes the birthplace for perhaps 400 chips.

In the "planar process" commonly used to create integrated circuits, these silicon wafers are coated with a layer of metal or silicon dioxide, a super insulator, and are then photolithographically patterned, using the following processes.

Photomask

When the magnetic tape containing a complete circuit design comes into the MOS wafer fabrication plant, it goes to the mask shop. The mask shop then produces a negative of the circuitry, at ten times its final size, on photosensitized glass. One four-inch square of glass, called a reticle, contains one layer of circuitry, so in a complex circuit they might

have to create, say, four or five different reticles.

Each reticle is then used to create a "mask" for that layer. Using another photographic process, the circuitry on the reticle is reduced to its final size (usually about a quarter of an inch on a side or smaller) and through a "step and repeat" process, reproduced over and over again on another piece of photosensitive glass, until the final mask contains several hundred exact reproductions of that layer of the circuit.

Now we get to the wafers. The final mask is inserted into a photolithographing machine along with a wafer that's been coated with a photosensitive material called photoresist. Light shines through the mask and exposes the photoresist, which hardens it. The parts of the photoresist that are *not* exposed, however (those areas where the mask blocks out the light), stay soft. The soft photoresist is removed, exposing the material underneath, which might be a metal or might be an oxide coating that's been put on the wafer, depending on the specs for that particular circuit. Thousands of wafers might go through this process for each layer of circuitry.

As you can guess, when you're working with microscopic circuits, any slight change in conditions has a devastating effect and a particle of dust is like a boulder. So all this (and most of the remaining processes) takes place in a "clean room", where temperature is controlled to within one degree Fahrenheit, humidity is held constant and particles are almost completely filtered out—the perfect work environment for anyone with allergies. People who work in these clean rooms wear special coveralls, booties and hair coverings. Even mustaches and beards must be covered.

Wafers are inspected under black light prior to processing.

Plasma Etching, Diffusion, Ion Implantation

Wherever the photoresist on a wafer has been removed in the photolithographic process it will be exposed, in the next step, to a "plasma" or excited chemical vapor to achieve a desired effect. In the past, liquids rather than vapors were used in this step, but the areas to be etched eventually became so small that liquids couldn't get into them anymore.

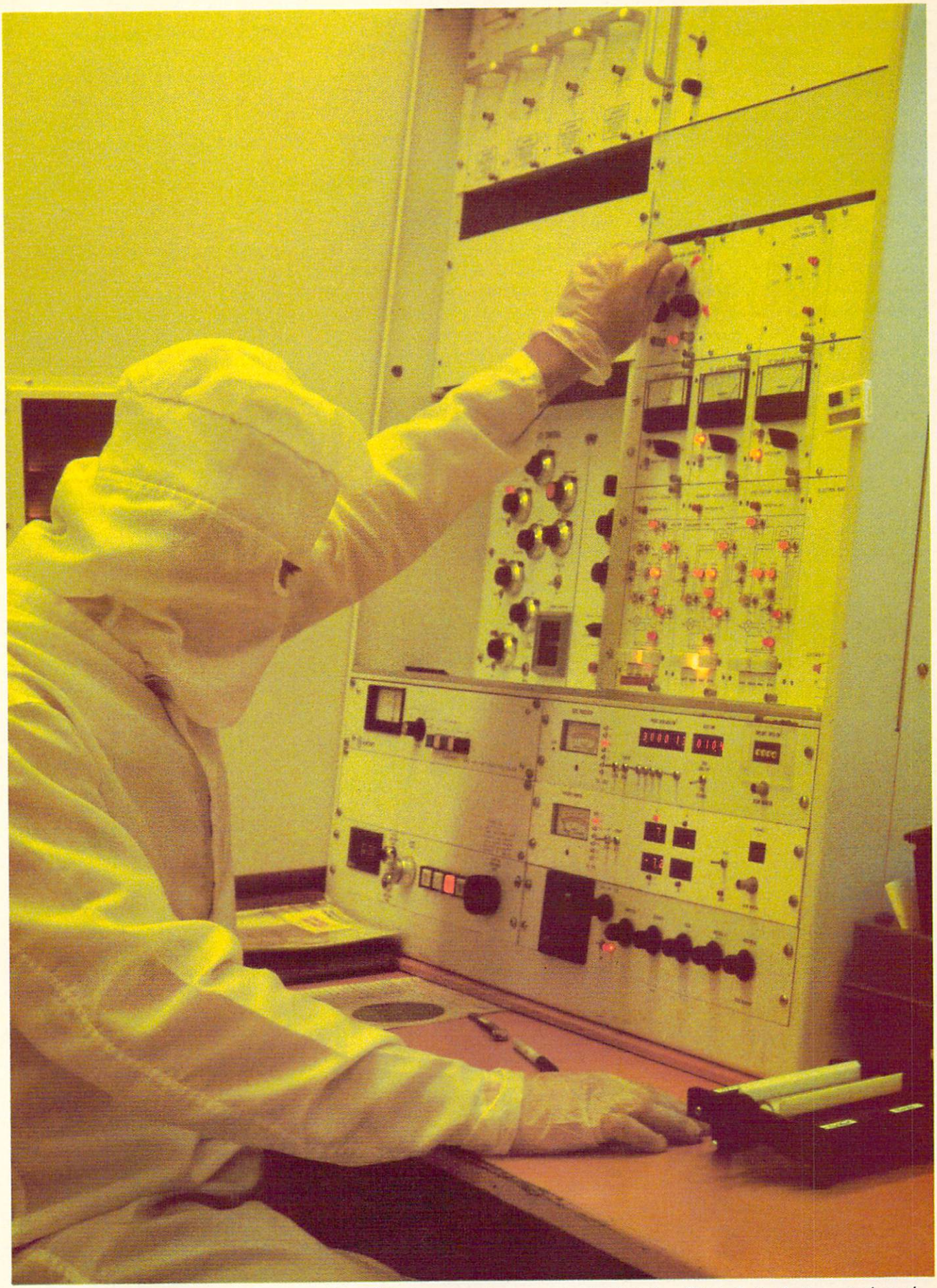
Depending on the specific circuit, the wafer may then go to either the diffusion furnace or to ion implantation. This is where impurities like boron, arsenic or phosphorus are added to create layers of semiconductors. Which chemicals are used and when, where they are placed on the wafers and the method used to place them are all determined by the needs of the individual circuit design.

One method of doping, or adding impurities, is with a diffusion furnace, which operates at temperatures up to about 1000 degrees centigrade. Wafers are loaded into quartz racks and advanced at a programmed rate into the furnace. The appropriate chemical gases are then circulated in a specific order, called a "recipe" (the language used in these processes sure is food oriented, isn't it?), to achieve a desired effect.

For newer, more complex circuits the diffusion process is sometimes not the best method. In that case, ion implantation is used. The ion implantation equipment does just what you think it does—it shoots individual ions of the appropriate chemical into the silicon. The number of ions and the depth at which they are implanted are crucial and are carefully controlled, which is probably an understatement.

Capping it Off

After about three weeks of going back and forth among these various processes, the wafers are finally finished. They're then coated with a protective layer of silicon nitride (known as SINCAP) and go through one more photolithographic process to expose the tiny connector pads around each chip (they won't be much good if



Newer, more complex circuits are sometimes "doped" using extremely accurate ion implantation. This is the implanter control panel.

electricity can't get in and out).

Before the wafers get cut up into individual chips they go through a final test called "wafer sort". In this process, tiny (and I mean tiny) mechanical "fingers" make connection with each individual chip on a wafer. The fingers are connected to a computer that determines whether the chip is working properly. If it isn't, the chip is marked with a little dot of ink. Industry-wide, the yield of good chips is usually less than 50%, so you can see why it is important to find the bad ones *before* you go through all the cutting and packaging processes.

Packaging

Finally the wafers are cut up into individual chips using an extremely precise diamond saw and are wired into their DIPs (remember the caterpillars?). Of course, there's wiring like you normally think of it and then there's wiring chips—with gold wire finer than an infant's eyelash, under a microscope.

At last the lids are put on if it's a ceramic DIP—or the plastic is injection-molded around the chip if it's not—and you've got yourself a completed "device", ready to go into your computer—or dishwasher. Just hope they don't confuse the two, or you'll end up with a rebellious dishwasher that entertains your kids and calculates your water bill instead of scrubbing pots.

It's almost a cliche by now, but I'm going to say it anyway. In the future, you can bet that IC's will continue to get smaller and more complex, so you will be able to do a whole lot more with much less. For instance, new discoveries that increase photographic resolution continue to influence the processes used in chip making. The size of circuits, after all, is determined at least in part by how fine the resolution can be when you're making a photomask. And as computers design new computers, the cumulative effect could be staggering.

But you can read that kind of conjecture anywhere. I just wanted, mainly, to give you the facts. So there they are. Hopefully you won't ever have to sink ashamedly into the shadows again when your techie friends start to talk about things like semiconductors, integrated circuits and chips.

References

National Geographic, October, 1982, offers a very detailed, yet eminently readable explanation of microchips and the surrounding technology, aimed at the layman.

Scientific American, September,

Important Dates in Microcomputer History

1642—Geared adding machine invented by French mathematician Blaise Pascal (then 19 years old). Later improved upon by Gottfried Leibniz.

1820's—"Difference Engine" created by Charles Babbage in England to calculate precise astronomical tables for navigational use.

1830's—"Analytical Engine" designed by Charles Babbage. It was to have a memory that could hold a thousand fifty-digit numbers and a "mill" to perform arithmetic operations. He proposed that information be fed into the Analytical Engine on punched cards, like those used at the time in advanced weaving looms. Although the Analytical Engine was beyond the scope of metal work of the time and could not be constructed, Babbage is still considered the father of the computer era.

1880's—Fast punched-card data system invented by Herman Hollerith to help the U.S. Census Bureau process data collected in the 1880 census. Paved the way for electronic sorting of data.

1884—"Comptometer Business Machine" built by Dorr E. Felt, using meat skewers, rubber bands, staples and a wooden macaroni box. It performed general purpose multiplication and division and eventually became one of the most important business machines of the early 20th century.

Early 1900's—First use of crystal radio detectors (early point-contact diodes, which used semiconductors).

1902—Development of the vacuum tube by Ambrose Fleming.

1944—The first "real" computer, Harvard's Mark I automatic general purpose digital calculator, developed by IBM in conjunction with Harvard. Howard Aiken, one of its developers, called it "Babbage's dream come true."

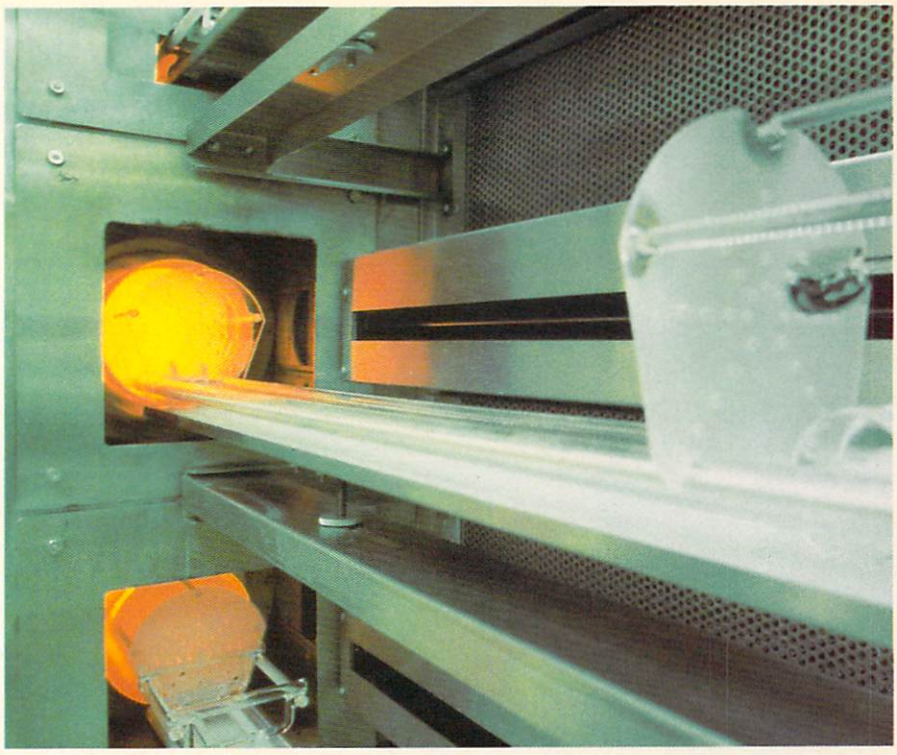
1945—Development of ENIAC (Electronic Numerical Integrator and Calculator), the first digital electronic computer. It weighed 30 tons, occupied a room 30 feet by 50 feet and contained 18,000 vacuum tubes. The only way to change a program was to rewire the machine. Its developers believed that about four of these machines could handle the computational needs of the entire United States.

1945—EDVAC (Electronic Discrete Variable Automatic Calculator) introduced by John von Neumann of Princeton University. The first to use on-off switches to represent instructions and data, and the first to allow both data and the program itself to be stored in memory.

1947—Creation of the first commercial semiconductor, the bi-polar transistor, by William Shockley, Walter Brattain and John Bardeen at Bell Labs.

1977, contains a more technical, maybe not so readable, but very useful explanation of microelectronics for the more engineering-minded. *Technology Illustrated,* February/March, 1982, presents somewhat superficial but nevertheless helpful information on chips. Workman Publishing's *1984 Computer Desk Diary* is a good place to find interesting tidbits about computing (like the source of the word "robot") and photos (like one of the original ENIAC computer).

For even more, check your library's *Reader's Guide to Periodical Literature,* starting about 1972—especially the articles that appeared in magazines like *Popular Electronics* and *Popular Science.* Some of them, I might add, are good for a laugh, as well as a source of information. C



The diffusion furnace, where wafers receive selective "doping" with impurities like boron, phosphorus or arsenic.

1951—Development of printed circuits by Danko and Abrahamson of the U.S. Army Signal Corps.

1951—Jay Forrester patents first core memory, allowing computers to have fast random access storage of large amounts of information.

1952—Possibility for developing semiconductor integrated circuits first perceived by G.W.A. Dummer of the Royal Radar Establishment in England.

1955—First field-effect transistors built and tested. But unstable fabrication processes, resulting in poor yields, prevented mass production.

1956—First computer kit marketed by Heath Company. The cost was $700.

1957—Development of the planar process by Robert Noyce and Gordon Moore at Fairchild Semiconductor, based on work by Frosch and Derrick at Bell Labs and others.

1959—Development of the first integrated circuits as we know them (in "chip" form) at Fairchild Semiconductor and Texas Instruments.

1961—Robert Noyce awarded the patent for the integrated circuit. The world hardly notices.

1962—First practical fabrication of the MOSFET (metal oxide field-effect transistor) and development of the MOS integrated circuit.

1967—Major MOSFET fabrication problems solved, resulting in increased chip yields and reduced costs.

1968—First MOS memory chips, containing 64 to 256 *bits* of random access memory.

1970—First MOS calculator chips introduced. First 1-kilobit (about 128 bytes) fully decoded RAM introduced.

1971—First central processing unit (CPU) on a single microchip developed by Ted Hoff at Intel. Paved the way for the first microcomputers, which were developed shortly thereafter.

1972—First general-interest magazine article about microcomputers: *Business Week,* May 12, "Microcomputers Aim at Huge New Market".

1974—6502 microprocessor developed at MOS Technology, Inc.

1975—Commodore acquires MOS Technology, Inc.

1977—First PET microcomputers marketed, containing 8K RAM.

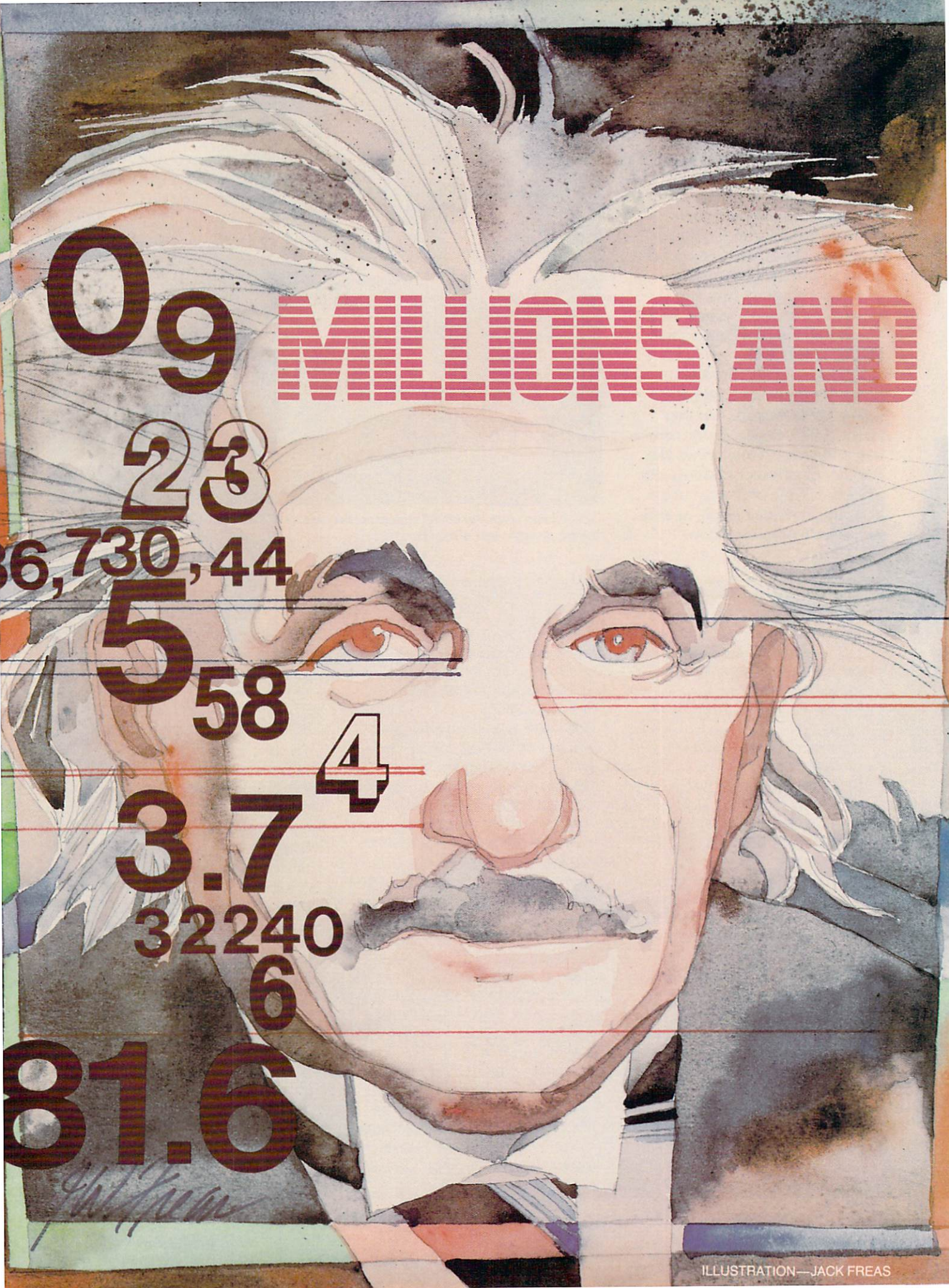
1980—Improved fabrication processes allow individual chips to contain up to 60,000 transistors.

1981—First easily affordable, full-featured home computer, the VIC 20, introduced by Commodore.

1982—Introduction of the Commodore 64, with more capabilities than many expensive business systems, at a low consumer price.

1983—Microcomputers become as affordable and as common as many home appliances.

1984—The best is yet to come. C



ILLUSTRATION—JACK FREAS

Thanks to modern technology, your computer deals with measurements on the scale of millions. There's a dot clock that ticks over eight million times a second and chips with components no larger than 50 millionths of a meter. In this article we'll look at the scales of both time and size as they relate to the 6502 chip.

THE MICROCHIP

By Jim Gracely

Big numbers are both mysterious and fascinating. Small numbers are just about the same as big numbers. When we want to describe something bigger than anything else we just use a number like "million": "You'll never guess in a million years", or "There must be a million of them", are both common statements. Once a number gets big enough, we lose any concept of how much it really is.

Here is an interesting example to test your skills in large numbers. Without the use of a calculator (or computer) choose which of the following represents the most money.

1. Two tons of gold (market price as of October 11, 1983)?
2. A dime for every inch between New York city and Los Angeles?
3. A nickel for every yard between the earth and the moon?

In this example, the scale of units involved is much greater than anything we're used to dealing with. The largest amount above is $25,280,000. I'll tell you which one adds up to that later. How much is twenty five million dollars? Well, if you used it to buy gasoline, you could fill a car with a 15 gallon tank every day for 3802 years!

Time

How does this relate to microcomputers? The clock crystal in the VIC and 64 has a frequency of 1Mhz. This means that it "ticks" one million times a second. Many machine language commands (such as INX and CLC) are executed in two ticks! That means that they take two millionths of a second to execute.

Now we're going to change the scale of the computer's clock and make each tick one second (multiply the clock by a million). This way we can more readily visualize some of the timing. Now, what used to take the computer one second to perform takes 278 hours or 11.5 days. Here is a chart that compares some "real time" events to the number of ticks and length of time on our new clock.

Real Time Event	How Many Ticks of a 1Mhz. Clock	Length of Time if 1 Tick = 1 Second
1 millionth of a second	1	1 second
1 second	1,000,000	278 hours or 11.5 days
1/60 of a second	16,667	4.6 hours
One "flap" of a house fly's wing. (according to Isaac Asimov—see Bibliography)	3160	53 minutes
Adding 5 and 7 in machine language		
LDA #5	2	2 seconds
CLC	2	2 seconds
ADC #7	2	2 seconds
STA 251	3	3 seconds
TOTAL	9 millionths of a second	9 seconds

This helps to bring a little insight into the speed of the microprocessor. Here is another question for you: If you were to work for 11.5 days straight with only a half hour break every four and a half hours would you consider yourself overworked? This is the analogy to the interrupt routine of the VIC and 64 which occurs 60 times a second. The 4.5 hours is the time spent running your basic program and the half hour break is the interrupt routine.

How long does it take to run a program? Well that depends on what the program does. Using our same expanded clock, let's look at a simple addition problem in BASIC. The addition 5+7 is performed in about .5 milliseconds. This is 500 ticks of the clock or 500 seconds (eight minutes) on our "upscale" clock. Look back at the chart to the short machine language program. That program also added five and seven, but it took only nine seconds. Dividing 500 by nine we find that the machine language program was over 50 times faster! Even going back to the 500 seconds it took in BASIC, eight minutes out of 11.5 days is still pretty good. In fact we could add five and seven about 2000 times in a million seconds.

We've spent enough time on this subject, so onward. Oh, that's right, about that question at the beginning. If you selected C, you were not correct. The average distance to the moon is 238,900 miles or 420,464,000 yards. At a nickel a yard it comes to $21,023,200. If you guessed B you were also not correct. The distance from New York city to Los Angeles is 2794 miles. This is 14,752,320 feet or 177,027,840 inches. At one dime on the inch this is $17,702,784. If you guessed A you were correct!! Congratulations!! On the day I put this quiz together, the price of gold was $395 an ounce. Two tons is 4000 pounds or 64,000 ounces. At $395 per ounce this is $25,280,000.

Size

We are now going to turn our attention to the size of the 6502 chip. Most people have seen a chip case before (what some call a "caterpillar"). The logical deduction then is to say "so that's a 6502 chip". Well that's not entirely correct. The black plastic or ceramic case with 28 pins ("legs") is actually a container for the 6502 chip. The chip itself is a little silver rectangle (a piece of silicon) about three by four millimeters. This is about the size of the little graphic boxes on the front of your VIC or 64 keys (if you own a PET/CBM computer it's just about the size of the cursor).

However, a little silver square does not a microprocessor make! There must be something on that square you say? Well indeed there is. In fact, there are over 5000 "somethings" on that square. A "something" is known in the electronics world as a component. A component may be a transistor, a resistor or a capacitor (there are others and variations but these are the most common).

Now let's look at the scale of a component that is so small that 5000 will fit on a single chip. If the 6502 chip is three by four millimeters and we divide this down into 5000 pieces, each piece would be about 50 micrometers on each side. This is 50 millionths of a meter. To give you an idea of how small this is we'll compare it to a grain of sand. According to Isaac Asimov (again) a typical grain of sand is about 200 micrometers on a side. We could put 16 of our components on one face of that grain! For those inland-lubbers, the eye of a needle is about one by .25 millimeters (according to Gracely—I got some strange looks trying to measure that!). We could pack 100 of our components in that space!

What if we wanted to enlarge the component to something of a good size—say a football field. A football field is roughly 100 meters long, so we would need to enlarge our component two million times. How big is the little silver chip now? Well it's two million times bigger or about 3.5 miles by five miles. What about the case or caterpillar? It's normally about two inches long and half an inch wide, but now... it's 64.5 miles long and 17 miles wide. In fact, if we let it sit on its pins or legs it would be almost ten miles high. To take the analogy one step further, how big would the VIC be now? 512 miles wide by 236 miles deep by 74 miles high.

Amazing as this may seem, keep in mind that the 6502 is not a very "dense" chip. In other words, 5000 components on a chip is not that many. On the front edge of technology, with 256,000-bit memory chips, more than half a million components may be on one chip. Going back to our eye of the needle, we could put about 2500 of these components into that space!

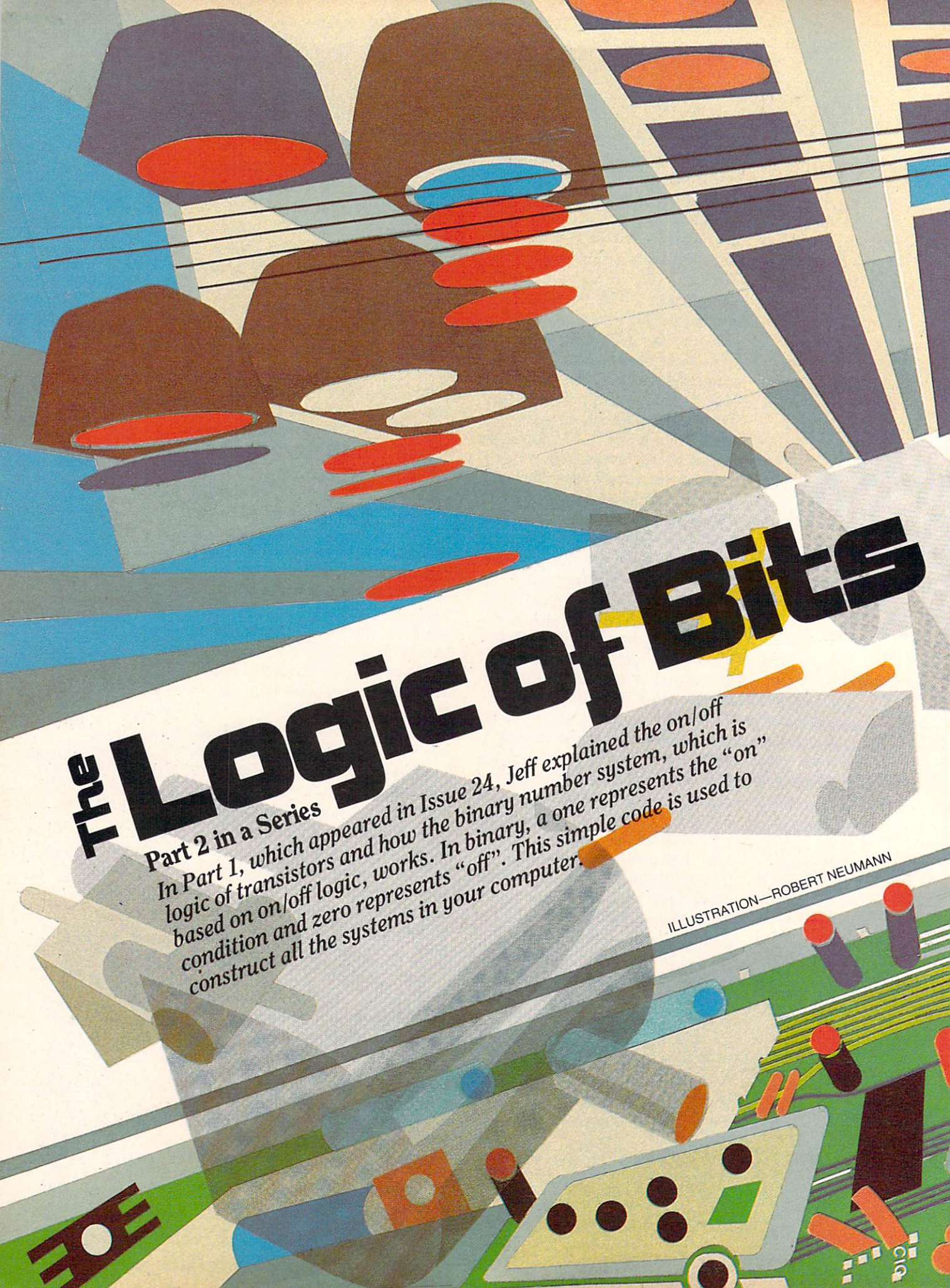
Well, that's our little trip through the time and size of the 6502 chip. I realize that this kind of article isn't packed full of useful facts and figures, but that's okay. It is important to stop now and then and think about what you're actually working with. An understanding and appreciation of any machine allow you to work with it properly, use it to its fullest extent and at the same time recognize its limitations.

Bibliography

The Measure of the Universe, Isaac Asimov. Harper & Row Publishers, New York: 1983.

Microprocessors And Microcomputers, Ronald J. Tocci and Lester R. Laskowski. Prentice-Hall, Inc., Englewood Cliffs, New Jersey: 1979.

C



The Logic of Bits

Part 2 in a Series

In Part 1, which appeared in Issue 24, Jeff explained the on/off logic of transistors and how the binary number system, which is based on on/off logic, works. In binary, a one represents the "on" condition and zero represents "off". This simple code is used to construct all the systems in your computer.

ILLUSTRATION—ROBERT NEUMANN

and Pieces

By Jeff Hand

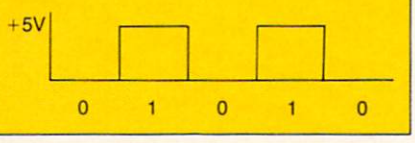
It's amazing that the simplest of concepts can generate the most complex of systems. The genetic code, for instance, which generates all the myriad of life on earth and determines everything about us, is composed of only four compounds. All of the cosmos is made up of only a little over 100 elements. Similarly, the computer is based on the simple concept of on/off and a few logic gates, yet look at the complex systems that have been developed. This part of our series will help you unravel some of that complexity.

In Part 1 we saw how the concept of the on/off transistor was developed into the binary, decimal and hexidecimal number systems used in computers. This part will show you how the same on/off (two-state) transistor concept develops into two-state logic and logic circuits and how those concepts are used in

software and hardware applications. Along the way we will pick up some other computer concepts like serial and parallel data transmission, Boolean algebra and encoders/decoders.

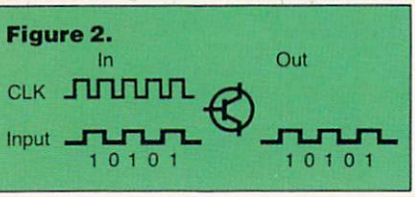
As we saw in Part 1, a transistor acts as a switch—it is either on or off. Figure 1 shows a signal, changing over time, that is fed into a transistor, as if you were turning the kitchen light on and off several times. If you remember, Commodore computers are TTL machines—that is, they use transistor-transistor logic—and they use five volts for "on" and zero volts for "off".

Figure 1.



This changing signal can be fed into a transistor as shown in Figure 2. The clock signal is a continuous pulse signal that is used as a timing reference for activities in the computer. In the Commodore computer there are a million clock pulses in one second. I'll explain the importance of the clock signal in a later article, but for now I've mentioned it just to facilitate your understanding of serial transmission.

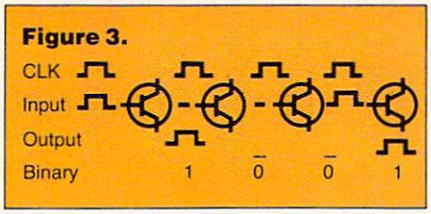
Figure 2.



In Figure 2 the information is leaving the transistor in a serial fashion. That means the bits of a word are sent out one after another through a single line, which is kept in sync with the rest of the computer by the clock signal.

Besides the serial method of moving data, there is a parallel method. Parallel transmission occurs when all the bits of a word are moved at the same time over different lines as shown in Figure 3.

Figure 3.



Figures 2 and 3 illustrate the basic difference between serial and parallel information transfer. Now when you hear about a serial interface, like an RS-232 for instance, where the information moves through one bit at at time, you should have an idea of how that method of information transfer compares to a parallel interface.

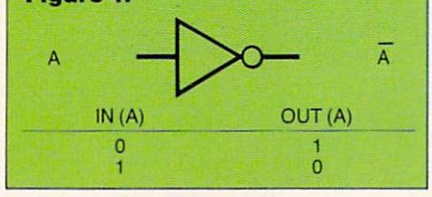
These signals can represent many types of information. The digital signal can mean one bit of a binary number or one bit of a binary code such as CBM ASCII, or it can be the control signal of an external event coming across an I/O port.

A logic gate operates on one or more inputs and produces one output signal. Logic gates are an arrangement of electronic components like transistors and resistors that act in certain ways on the incoming signals to create the output. Let's take a look at these various components and what they do.

Inverters

An inverter has only one input. The output signal is always the opposite of the input signal. If a one (five volts) enters the inverter, the output is zero. Conversely if a zero enters, the output is one. The inverter is also known as a NOT gate. Figure 4 shows a schematic representation of the inverter, and includes a logic table.

Figure 4.



The logic table is more commonly known as a truth table. The truth table is a listing of all the input possibilities and the associated output for the logic gate. The inverter is easy enough to remember without a truth table but other logic gates can become confusing, which is when the truth table becomes invaluable, as you will soon see.

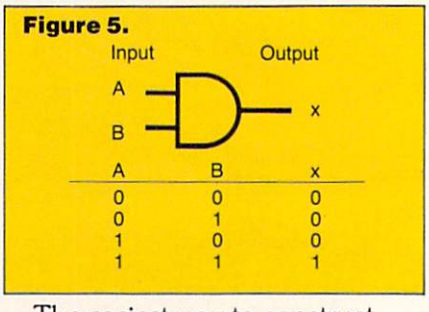
Boolean algebra is one of the major tools in understanding logic gates and digital computers. It was developed about 1854 by George Boole, a mathematician. It was little used until engineers began using it to design switching relays for telephones. Then, with the advent of computers, Boolean algebra became indispensable for understanding computer logic.

There is nothing difficult about Boolean algebra and as I explain each logic gate, I'll also give you the Boolean symbolic representations for manipulation. For instance, Boolean algebra represent the NOT gate we just discussed with a line over the input. So if the input to an inverter is A then the output is represented by $\overline{A}$.

AND Gate

The distinguishing characteristic of the AND gate is that all input signals must be high (five volts) to produce a high (five volts) output. Figure 5 shows a schematic of the gate and the associated truth table.

Figure 5.



A	B	x
0	0	0
0	1	0
1	0	0
1	1	1

The easiest way to construct the truth table is to count in binary, starting with zero, until all the input values have been covered. Next, place the corresponding output on the righthand side.

Logic gates can have more than two inputs. A two-input gate will generate 2↑2 or four possibilities. A three-input gate will generate 2↑3 or eight possible conditions. An easy way to know the total

number of possibilities for a given number of inputs is to use the formula 2↑N, where N is the total number of inputs to the logic gate. If you have ten inputs on the gate, the total number of possibilities is 2↑10 or 1024, which is an awfully big truth table for an AND gate, but all you have to remember is the only input combination that will generate a high output is 11 1111 1111.

For the purpose of explaining Boolean algebra, the inputs to the AND gate are labeled A,B,C and so on and the outputs are labeled x. This will also be true in the rest of the gates we'll be discussing.

The Boolean function for an AND gate is the same as normal multiplication. But what makes this algebra so easy is that we have only a one or a zero to multiply with. Therefore our answers can be only zero or one. The Boolean equation can be written in a couple of ways:

A AND B = x
or
A • B = x

The second version is an abbreviated form, in which the dot is the same operator as in multiplication. These are just two ways of expressing the same AND gate function.

Solving these equations is so easy you're going to wonder why you didn't already know this stuff. Let's go through the AND truth table using Boolean algebra and see if we come up with the same conclusion.

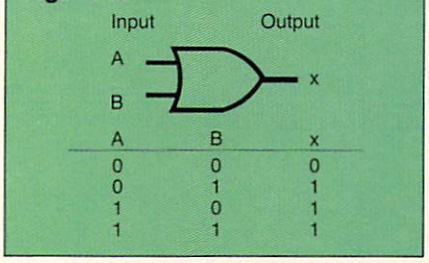
A • B = x
0 • 0 = 0
0 • 1 = 0
1 • 0 = 0
1 • 1 = 1

You can see that all you have to do is be able to multiply by zero or one. Not too hard. Boolean algebra will simplify your task for what appears to be a tangled mess of logic gates. You can always reduce the mess to Boolean equations of simple multiplication for the AND gates and addition for the OR gate we'll discuss next.

OR Gate

The OR gate will produce a high output for any high input. Figure 6 shows a schematic and truth table for the OR gate.

Figure 6.



A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

In Boolean algebra the OR gate is equivalent to addition. Here is the general operation:

A OR B = x

This can be rewritten in the Boolean equivalent using a plus sign:

A + B = x

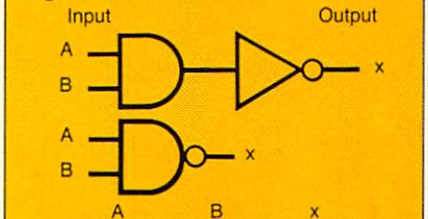
We can walk through this equation with various input possibilities and see if we come up with the same results as in the truth table.

A + B = x
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

What might confuse you here is how one plus one can equal one. It's because we're working with two-state logic, in which the only numbers are zero and one. It's just a convention that 1 + 1 = 1 in Boolean algebra.

The AND and OR gates, along with the inverter, are the major building blocks from which the computer and all of its capabilities are fundamentally built. To give you a little more background, however, we'll look at several more gates in Figures 7, 8 and 9, although these are actually combinations of the three gates we've just discussed.

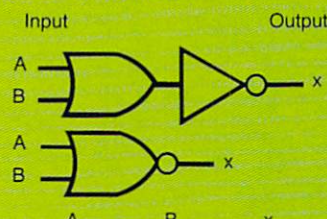
Figure 7. NAND Gate



A	B	x
0	0	1
0	1	1
1	0	1
1	1	0

Boolean Equation: A • B = x

Figure 8. NOR Gate

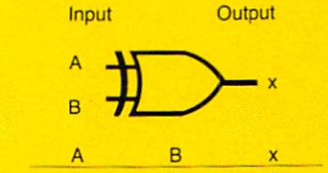


A	B	x
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Equation: A + B = x

The EX-OR gate in Figure 9 has a high output only when the input levels are different. If the input voltages are the same, then the output is zero. For the EX-OR gate there can be only two input values.

Figure 9. EX-OR (Exclusive OR Gate)



A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Equation: A + B = x

To simplify a generally complex definition, a decoder takes the binary information and usually translates up toward something that mere humans can easily understand. An encoder, on the other hand, takes information and translates it down into the machine's level of ones and zeros.

Figure 10 shows a general diagram of a binary-to-decimal one-of-sixteen decoder.

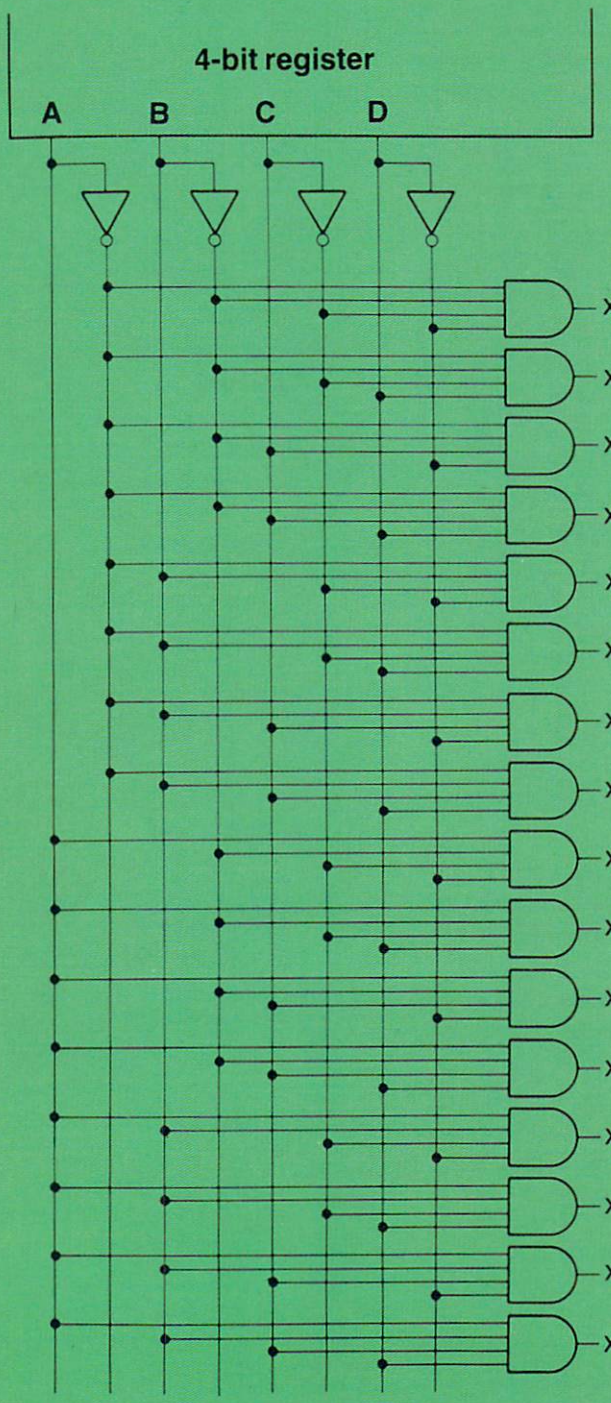
Now you can use your knowledge of logic gates and Boolean

algebra to figure out how this decoder works. The binary input signals are transported through the lines connected to A B C D. The signal then travels through both the red and green lines at the same time. At each point where there is a dot on either red or green, that signal is then fed into the AND gate. Looking back to the truth table for the AND gate, you find only all high-input signals will cause the output signal to be high.

Let's walk through the first input together. Assume all zeros come through A B C D. Trace the lines through and zero goes through all the blue lines. The inverters change the signal in all the green lines to one. The only AND gate that has all high inputs is X_0, which is connected to a display that shows a zero has come through somewhere deep in the computer. We can walk through the entire decoder this way. The Boolean algebra equivalents are also filled in for you.

An encoder is shown in Figure 11. This is a hexidecimal encoder.

Figure 10. Binary-to-Decimal Decoder



Boolean	Binary Input	Decimal Output
$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$	0000	0
$\overline{A}\,\overline{B}\,\overline{C}D$	0001	1
$\overline{A}\,\overline{B}C\overline{D}$	0010	2
$\overline{A}\,\overline{B}CD$	0011	3
$\overline{A}B\overline{C}\,\overline{D}$	0100	4
$\overline{A}B\overline{C}D$	0101	5
$\overline{A}BC\overline{D}$	0110	6
$\overline{A}BCD$	0111	7
$A\overline{B}\,\overline{C}\,\overline{D}$	1000	8
$A\overline{B}\,\overline{C}D$	1001	9
$A\overline{B}C\overline{D}$	1010	10
$A\overline{B}CD$	1011	11
$AB\overline{C}\,\overline{D}$	1100	12
$AB\overline{C}D$	1101	13
$ABC\overline{D}$	1110	14
$ABCD$	1111	15

It takes the information, in this case from a keypad on a calculator or computer, and converts that information into a binary form the computer can understand. This is a very simple encoder for illustration and learning purposes, but the phenomenon is the same when you hit a key on your computer, except the encoder is much larger.

If we look back at the OR gate truth table we see that only high input will produce a high output. For example if we hit the "A" key we can see that the A switch goes down and five volts is sent down the line. This produces a high signal in the X3 or X1 OR gates, which translates to 1010 in binary or a ten in decimal. This gate goes only to hexidecimal Fa (decimal 15, which is actually 16 including 0), but a normal eight-bit word can generate 256 different combinations of decimal or FF(hx) possibilities. You can see the matrix gets rather large and cumbersome to use.

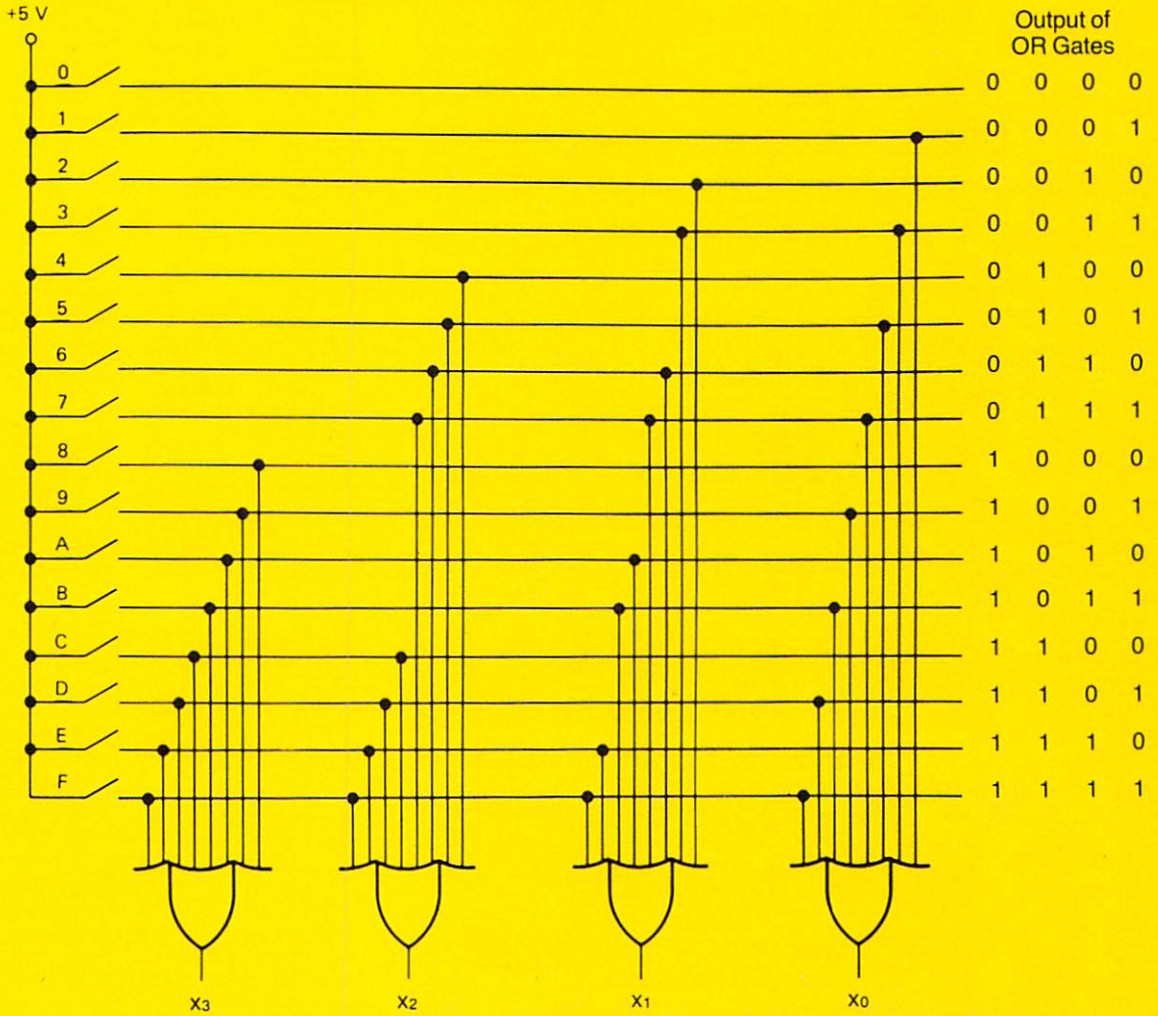
The encoder and decoder should give you some idea of the building blocks used in computer hardware. Now we will turn our attention to how logic can be used in software applications.

In programming, logical operators generally are used to test for a bit within a byte or a particular occurrence. For example we could store information coming through the I/O port. Say you have to keep the temperature in a room above 80 degrees. If the temperature stays above 80 your thermometer interface puts zeros in the I/O port. But if the temperature drops below 80, a one is placed in the I/O register.

The program would then take the information in the I/O register and place it into, say, register 400. Your next step is to take the contents of register 400 and test it. "AND" the contents of register 400 with a 1. Out of all the combinations possible the result will be zero, except if there is a one in register 400. From here your program can branch to get the temperature back up to 80 degrees in the room.

Part 3 in our series will explain two's complement and simple math operations. **C**

Figure 11. Hex-to-Binary Encoder



6502 OP-CODES

Condensed By Jim Gracely

We get a lot of questions concerning the op-codes for the 6502. People buy the VIC or 64 Programmer's Reference Guide *and see all the tables of op-codes. However, these guides don't explain what the op-codes do or how to use them. One very important piece of information about each op-code is how it effects the processor status register. In fact, even some of the textbooks on machine language lack this information. Here then is a list of all the 6502 op-codes with a short description of what they do and how they effect the status register. By the way, for those of you who may be wondering, the 6510 in the 64 uses the same set of op-codes as the 6502.*

This information was condensed by Jim Gracely from the *MCS6500 Microcomputer Family Programming Manual* published by MOS Technology, Inc. Unfortunately this book is no longer in print, although you may find copies of it in your local public or college library.

LDA: Load Accumulator with Memory

When instruction LDA is executed by the microprocessor, data is transferred from memory to the accumulator and stored in the accumulator.

LDA affects the contents of the accumulator, does not affect the carry or overflow flags; sets the zero flag if the accumulator is zero as a result of the LDA, otherwise resets the zero flag; sets the negative flag if bit seven of the accumulator is a one, otherwise resets the negative flag.

STA: Store Accumulator in Memory

This instruction transfers the contents of the accumulator to memory. It affects none of the flags in the processor status register and does not affect the accumulator.

ADC: Add Memory to Accumulator with Carry

This instruction adds the value of memory and carry from the previous operation to the value of the accumulator and stores the result in the accumulator.

This instruction affects the accumulator; sets the carry flag when the sum of a binary add exceeds 255 or when the sum of a decimal add exceeds 99, otherwise carry is reset. The overflow flag is set when the sign or bit seven is changed due to the result exceeding +127 or −128, otherwise overflow is reset. The negative flag is set if the accumulator result contains bit seven on, otherwise the negative flag is reset. The zero flag is set if the accumulator result is zero, otherwise the zero flag is reset.

SBC: Subtract Memory from Accumulator with Borrow

This instruction subtracts the value of memory and borrow from the value of the accumulator, using two's complement arithmetic, and stores the result in the accumulator. Borrow is defined as the carry flag complemented; therefore, a resultant carry flag indicates that a borrow has not occurred.

This instruction affects the accumulator. The carry flag is set if the result is greater than or equal to zero. The carry flag is reset when the result is less than zero, indicating a borrow. The overflow flag is set when the result exceeds +127 or −127, otherwise it is reset. The negative flag is set if the result in the accumulator has bit seven on, otherwise it is reset. The Z flag is set if the result in the accumulator is zero, otherwise it is reset.

AND: Memory with Accumulator

The AND instructions transfer the accumulator and memory to the adder which performs a bit-by-bit "AND" operation and stores the result back in the accumulator.

This instruction affects the accumulator; sets the zero flag if the result in the accumulator is zero, otherwise resets the zero flag; sets the negative flag if the result in the accumulator has bit seven on, otherwise resets the negative flag.

ORA: "OR" Memory with Accumulator

The ORA instruction transfers the memory and the accumulator to the adder which performs a binary "OR" on a bit-by-bit basis and stores the result in the accumulator.

This instruction affects the accumulator; sets the zero flag if the result in the accumulator is zero, otherwise resets the zero flag; sets the negative flag if the result in the accumulator has bit seven on, otherwise resets the negative flag.

EOR: "Exclusive OR" Memory with Accumulator

The EOR instruction transfers the memory and the accumulator to the adder which performs a binary "Exclusive OR" on a bit-by-bit basis and stores the result in the accumulator.

This instruction affects the accumulator; sets the zero flag if the result in the accumulator is zero, otherwise resets the zero flag; sets the negative flag if the result in the accumulator has bit seven on, otherwise resets the negative flag.

SEC: Set Carry Flag

This instruction initializes the carry flag to a one. This operation should normally precede a SBC loop. It is also useful when used with a ROL instruction to initialize a bit in memory to a one.

This instruction affects no registers in the microprocessor and no flags other than the carry flag which is set.

CLC: Clear Carry Flag

This instruction initializes the carry flag to a zero. This operation should normally precede an ADC loop. It is also useful when used with a ROL instruction to clear a bit in memory.

This instruction affects no registers in the microprocessor and

no flags other than the carry flag which is reset.

SEI: Set Interrupt Disable

This instruction initializes the interrupt disable to a one. It is used to mask interrupt requests during system reset operations and during interrupt commands.

It affects no registers in the microprocessor and no flags other than the interrupt disable which is set.

CLI: Clear Interrupt Disable

This instruction initializes the interrupt disable to a zero. This allows the microprocessor to receive interrupts.

It affects no registers in the microprocessor and no flags other than the interrupt disable which is cleared.

SED: Set Decimal Mode

This instruction sets the decimal mode flag D to a one. This makes all subsequent ADC and SBC instructions operate as a decimal arithmetic operation.

SED affects no registers in the microprocessor and no flags other than the decimal mode which is set to a one.

CLD: Clear Decimal Mode

This instruction sets the decimal mode flag to a zero. This causes all subsequent ADC and SBC instructions to operate as simple binary operations.

CLD affects no registers in the microprocessor and no flags other than the decimal mode flag which is set to a zero.

CLV: Clear Overflow Flag

This instruction clears the overflow flag to a zero. This command is used in conjunction with the set overflow pin which can change the state of the overflow flag with an external signal.

CLV affects no registers in the microprocessor and no flags other than the overflow flag which is set to a zero.

JMP: Jump to New Location

In this instruction, the data from the memory location located in the program sequence after the OP CODE is loaded into the low order byte of the program counter (PCL) and the data from the next memory location after that is loaded into the high order byte of the program counter (PCH).

The JMP instruction affects no flags and only PCL and PCH. Although the jump allows the change of program sequence, it does so without performing any test. So it is a JMP instruction that is employed when it is desired to change the program counter no matter what conditions have occurred.

BMI: Branch on Result Minus

This instruction takes the conditional branch if the N bit is set.

BMI does not affect any of the flags or any other part of the machine other than the program counter and then only if the N bit is on.

BPL: Branch on Result Plus

This instruction is the complementary branch to branch on result minus. It is a conditional branch which takes the branch when the N bit is reset (0). BPL is used to test if the previous result bit seven was off (0) and branch on result minus is used to determine if the previous result was minus or bit seven was on (1).

The instruction affects no flags or other registers other than the P counter and only affects the P counter when the N bit is reset.

BCC: Branch on Carry Clear

This instruction tests the state of the carry bit and takes a conditional branch if the carry bit is reset.

It affects no flags or registers other than the program counter and then only if the C flag is not on.

BCS: Branch on Carry Set

This instruction takes the conditional branch if the carry flag is on.

BCS does not affect any of the flags or registers except for the program counter and only then if the carry flag is on.

BEQ: Branch on Result Zero

This instruction could also be called "Branch on Equal." It takes a conditional branch whenever the Z flag is on or the previous result is equal to zero.

BEQ does not affect any of the flags or registers other than the program counter and only then when the Z flag is set.

BNE: Branch on Result Not Zero

This instruction could also be called "Branch on Not Equal." It tests the Z flag and takes the conditional branch if the Z flag is not on, indicating that the previous result was not zero.

BNE does not affect any of the flags or registers other than the program counter and only then if the Z flag is reset.

BVS: Branch on Overflow Set

This instruction tests the V flag and takes the conditional branch if V is on.

BVS does not affect any flags or registers other than the program counter and only when the overflow flag is set.

BVC: Branch on Overflow Clear

This instruction tests the status of the V flag and takes the conditional branch if the flag is not set.

BVC does not affect any of the flags and registers other than the program counter and only when the overflow flag is reset.

CMP: Compare Memory and Accumulator

This instruction subtracts the contents of memory from the contents of the accumulator.

The use of the CMP affects the following flags: Z flag is set on an equal comparison, reset otherwise; the N flag is set or reset by the result bit seven, the carry flag is set when the value in memory is less than or equal to the accumulator, reset when it is greater than the accumulator. The accumulator is not affected.

The purpose of the compare instruction is to allow the user to compare a value in memory to the accumulator without changing the value of the accumulator.

BIT: Test Bits in Memory with Accumulator

This instruction performs an AND between a memory location and the accumulator but does not store the result of the AND into the accumulator.

The bit instruction affects the N flag with N being set to the value of bit seven of the memory being tested, the V flag with V being set equal to bit six of the memory being tested and Z being set by the result of the AND operation between the accumulator and the memory if the result is zero. Z is reset otherwise. It does not affect the accumulator.

LDX: Load Index Register X From Memory

Load the index register X from memory.

LDX does not affect the C or V flags; sets Z if the value loaded was zero, otherwise resets it; sets N if the value loaded in bit seven is a one; otherwise N is reset, and affects only the X register.

LDY: Load Index Register Y From Memory

Load the index register Y from memory.

LDY does not affect the C or V flags, sets the N flag if the value loaded in bit seven is a one, otherwise resets N, sets Z flag if the loaded value is zero, otherwise resets Z, and only affects the Y register.

STX: Store Index Register X in Memory

Transfers value of X register to addressed memory location.

No flags or registers in the microprocessor are affected by the store operation.

STY: Store Index Register Y in Memory

Transfer the value of the Y register to the addressed memory location. STY does not affect any flags or registers in the microprocessor.

INX: Increment Index Register X by One

Increment X adds one to the current value of the X register. This is an eight-bit increment which does not affect the carry operation, therefore, if the value of X before the increment was FF, the resulting value is 00. INX does not affect the carry or overflow flags; it sets the N flag if the result of the increment has a one in bit seven, otherwise resets N; sets the Z flag if the result of the increment is zero, otherwise it resets the Z flag. INX does not affect any other register other than the X register.

INY: Increment Index Register Y by One

Increment Y increments or adds one to the current value in the Y register, storing the result in the Y register. As in the case of INX the primary application is to step through a set of values using the Y register. The INY does not affect the carry or overflow flags, sets the N flag if the result of the increment has a one in bit seven, otherwise resets N, sets Z if as a result of the increment the Y register is zero otherwise resets the Z flag.

DEX: Decrement Index Register X by One

This instruction subtracts one from the current value of the index register X and stores the result in the index register X.

DEX does not affect the carry or overflow flag, it sets the N flag if it has bit seven on as a result of the decrement, otherwise it resets the N flag; sets the Z flag if X is a zero as a result of the decrement, otherwise it resets the Z flag.

DEY: Decrement Index Register Y by One

This instruction subtracts one from the current value in the index register Y and stores the result into the index register Y. The result does not affect or consider carry so

that the value in the index register Y is decremented to zero and then through zero to FF.

Decrement Y does not affect the carry or overflow flags; if the Y register contains bit seven on as a result of the decrement the N flag is set, otherwise the N flag is reset. If the Y register is zero as a result of the decrement, the Z flag is set otherwise the Z flag is reset. This instruction only affects the index register Y.

NOTE: Decrement of the index registers is the most convenient method of using the index registers as a counter, in that the decrement involves setting the value N on as a result of having passed through zero and sets Z on when the results of the decrement are zero.

CPX: Compare Index Register X to Memory

This instruction subtracts the value of the addressed memory location from the content of index register X using the adder but does not store the result; therefore, its only use is to set the N, Z and C flags to allow for comparison between the index register X and the value in memory.

The CPX instruction does not affect any register in the machine; it also does not affect the overflow flag. It causes the carry to be set on if the absolute value of the index register X is equal to or greater than the data from memory. If the value of the memory is greater than the content of the index register X, carry is reset. If the results of the subtraction contain a bit seven, then the N flag is set, if not, it is reset. If the value in memory is equal to the value in index register X, the Z flag is set, otherwise it is reset.

CPY: Compare Index Register Y to Memory

This instruction performs a two's complement subtraction between the index register Y and the specified memory location. The results of the subtraction are not stored anywhere. The instruction is strictly used to set the flags.

CPY affects no registers in the microprocessor and also does not affect the overflow flag. If the value in the index register Y is equal to or greater than the value in the memory, the carry flag will be set, otherwise it will be cleared. If the results of the subtraction contain bit seven on the N bit will be set, otherwise it will be cleared. If the value in the index register Y and the value in the memory are equal, the zero flag will be set, otherwise it will be cleared.

TAX: Transfer Accumulator to Index X

This instruction takes the value from accumulator A and transfers or loads it into the index register X without disturbing the content of the accumulator A.

TAX only affects the index register X, does not affect the carry or overflow flags. The N flag is set if the resultant value in the index register X has bit seven on, otherwise N is reset. The Z bit is set if the content of the register X is zero as a result of the operation, otherwise it is reset.

TXA: Transfer Index X to Accumulator

This instruction moves the value that is in the index register X to the accumulator A without disturbing the content of the index register X.

TXA does not affect any register other than the accumulator and does not affect the carry or overflow flag. If the result in A has bit seven on, then the N flag is set, otherwise it is reset. If the resultant value in the accumulator is zero, then the Z flag is set, otherwise it is reset.

TAY: Transfer Accumulator to Index Y

This instruction moves the value of the accumulator into index register Y without affecting the accumulator.

TAY instruction only affects the Y register and does not affect either the carry or overflow flags. If the index register Y has bit seven on, then N is set, otherwise it is reset. If the content of the index register Y equals zero as a result of the operation, Z is set on, otherwise it is reset.

TYA: Transfer Index Y to Accumulator

This instruction moves the value that is in the index register Y to accumulator A without disturbing the content of the register Y.

TYA does not affect any other register other than the accumulator and does not affect the carry or overflow flag. If the result in the accumulator A has bit seven on, the N flag is set, otherwise it is reset. If the resultant value in the accumulator A is zero, then the Z flag is set, otherwise it is reset.

Some of the applications of the transfer instructions between accumulator A and index registers X, Y are those when the user wishes to use the index register to access memory locations where there

are multiple byte values between the addresses. In this application a count is loaded into the index register, the index register is transferred to the accumulator, a value such as five, seven, ten, etc., is added immediate to the accumulator and results stored back into the index.

This is known as subroutine nesting and is often encountered in solving complex control equations.

To correctly use the stack for this type of operation requires a jump to subroutine and a return from subroutine instruction.

JSR: Jump to Subroutine

This instruction transfers control of the program counter to a subroutine location but leaves a return pointer on the stack to allow the user to return to perform the next instruction in the main program after the subroutine is complete. To accomplish this, JSR instruction stores the program counter address which points to the last byte of the jump instruction onto the stack using the stack pointer. The stack byte contains the program count high first, followed by program count low. The JSR then transfers the addresses following the jump instruction to the program counter low and the program counter high, thereby directing the program to begin at that new address.

The JSR instruction affects no flags, causes the stack pointer to be decremented by two and substitutes new values into the program counter low and the program counter high.

RTS: Return From Subroutine

This instruction loads the program count low and program count high from the stack into the program counter and increments the program counter so that it points to the instruction following the JSR. The stack pointer is adjusted by incrementing it twice.

The RTS instruction does not affect any flags and affects only PCL and PCH.

PHA: Push Accumulator on Stack

This instruction transfers the current value of the accumulator to the next location on the stack, automatically decrementing the stack to point to the next empty location.

The Push A instruction only affects the stack pointer register which is decremented by one as a result of the operation. It affects no flags.

PLA: Pull Accumulator From Stack

This instruction adds one to the current value of the stack pointer and uses it to address the stack and loads the contents of the stack into the A register.

The PLA instruction does not affect the carry or overflow flags. It sets N if the bit seven is on in accumulator A as a result of instructions, otherwise it is reset. If accumulator A is zero as a result of the PLA, then the Z flag is set, otherwise it is reset. The PLA instruction changes content of the accumulator A to the contents of the memory location at stack register plus one and also increments the stack register.

TXS: Transfer Index X to Stack Pointer

This instruction transfers the value in the index register X to the stack pointer.

TXS changes only the stack pointer, making it equal to the content of the index register X. It does not affect any of the flags.

TSX: Transfer Stack Pointer to Index X

This instruction transfers the value in the stack pointer to the index register X.

TSX does not affect the carry or overflow flags. It sets N if bit seven is on in index X as a result of the instruction, otherwise it is reset. If index X is zero as a result of the TSX, the Z flag is set, otherwise it is reset. TSX changes the value of index X, making it equal to the content of the stack pointer.

PHP: Push Processor Status on Stack

This instruction transfers the contents of the processor status register unchanged to the stack, as governed by the stack pointer.

The PHP instruction affects no registers or flags in the microprocessor.

PLP: Pull Processor Status From Stack

This instruction transfers the next value on the stack to the processor status register, thereby changing all of the flags and setting the mode switches to the values from the stack.

The PLP instruction affects no registers in the processor other than the status register. This instruction could affect all flags in the status register.

RTI: Return From Interrupt

This instruction transfers from the stack into the microprocessor the processor status and the pro-

gram counter location for the instruction which was interrupted. By virtue of the interrupt having stored this data before executing the instruction and the fact that the RTI reinitializes the microprocessor to the same state as when it was interrupted, the combination of interrupt plus RTI allows truly reentrant coding.

The RTI instruction reinitializes all flags to the position they were at the time the interrupt was taken and sets the program counter back to its pre-interrupt state. It affects no other registers in the microprocessor.

BRK: Break Command

The break command causes the microprocessor to go through an interrupt sequence under program control. This means that the program counter of the second byte after the BRK is automatically stored on the stack along with the processor status at the beginning of the break instruction. The microprocessor then transfers control to the interrupt vector.

Other than changing the program counter, the break instruction changes no values in either the registers or the flags.

LSR: Logical Shift Right

This instruction shifts either the accumulator or a specified memory location one bit to the right, with the higher bit of the result always being set to zero, and the low bit which is shifted out of the field being stored in the carry flag.

The shift right instruction either affects the accumulator by shifting it right one or is a read/modify/write instruction which changes a specified memory location but does not affect any internal registers. The shift right does not affect the overflow flag. The N flag is always reset. The Z flag is set if the result of the shift is zero and reset otherwise. The carry is set equal to bit zero of the input.

ASL: Arithmetic Shift Left

The shift left instruction shifts either the accumulator or the address memory location one bit to the left, with the bit zero always being set to zero and the bit seven output always being contained in the carry flag. ASL either shifts the accumulator left one bit or is a read/modify/write instruction that affects only memory.

The instruction does not affect the overflow bit, sets N equal to the result bit seven (bit six in the input), sets Z flag if the result is equal to zero, otherwise resets Z and stores the input bit seven in the carry flag.

ROL: Rotate Left

The rotate left instruction shifts either the accumulator or addressed memory left one bit, with the input carry being stored in bit zero and with the input bit seven being stored in the carry flags.

The ROL instruction either shifts the accumulator left one bit and stores the carry in accumulator bit zero or does not affect the internal registers at all. The ROL instruction sets carry equal to the input bit seven, sets N equal to the input bit six, sets the Z flag if the result of the rotate is zero, otherwise it resets Z and does not affect the overflow flag at all.

ROR: Rotate Right

The rotate right instruction shifts either the accumulator or addressed memory right one bit with bit zero shifted into the carry and carry shifted into bit seven.

The ROR instruction either shifts the accumulator right one bit and stores the carry in accumulator bit seven or does not affect the internal registers at all. The ROR instruction sets carry equal to input bit zero, sets N equal to the input carry and sets the Z flag if the result of the rotate is zero; otherwise it resets Z and does not affect the overflow flag at all.

INC: Increment Memory by One

This instruction adds one to the contents of the addressed memory location.

The increment memory instruction does not affect any internal registers and does not affect the carry or overflow flags. If bit seven is on as the result of the increment, N is set, otherwise it is reset; if the increment causes the result to become zero, the Z flag is set on, otherwise it is reset.

DEC: Decrement Memory by One

This instruction subtracts one, in two's complement, from the contents of the addressed memory location.

The decrement instruction does not affect any internal register in the microprocessor. It does not affect the carry or overflow flags. If bit seven is on as a result of the decrement, then the N flag is set, otherwise it is reset. If the result of the decrement is zero, the Z flag is set, otherwise it is reset. **C**

Easy Spell 64

Reviewed by Mark Cornacchio

A spelling program you can use with Commodore's Easy Script 64 *word processor.*

I don't know about you, but I am a bad speller and typist. I am lazy, too. Therefore, *Easy Spell* is perfect for me.

What is *Easy Spell? Easy Spell* is the spelling package made to work with *Easy Script,* the word processing software for the Commodore 64. *Easy Spell* does what the name implies: it checks spelling of files created by *Easy Script.*

Easy Spell is loaded directly from *Easy Script* in the same manner a text file is loaded. Once it is loaded, press any key to get the options menu. There are seven options. The first option is entitled "Check Text File". This option checks the spelling of the requested text file. *Easy Spell* will highlight unrecognized words. The qualification for an unrecognized word is a word that is not in the *Easy Spell* Master Dictionary or User Dictionary. To correct spelling mistakes once they are found, use the editing procedures normally used in *Easy Script.* Sometimes it is desirable not to change the unrecognized word. Press the "F1" function key to ignore unrecognized words.

Options three through five let one print, delete and search dictionaries. The print and delete options only apply to the User Dictionary. But before discussing the User Dictionary, I would like to discuss the Master Dictionary. The Master Dictionary has 20,000 words. The Master Dictionary cannot be altered. Parts of the dictionary can be seen by using the search option. Pattern matching helps search the Master Dictionary quickly. Now! What we all have been waiting for—a word about the User Dictionary. *Easy Spell* may produce an unrecognized word that is actually spelled correctly. To prevent this from happening in the future, add the word to the User Dictionary. To add a word to the User Dictionary, simply press the left arrow key.

This software is a must for anybody who owns *Easy Script. Easy Spell* enhances *Easy Script* so immensely, in my opinion, that it makes *Easy Script* the best word processing package on the market for the Commodore 64. In addition to the features mentioned above, *Easy Spell* can "learn" words while correcting text. The manual has more information on "learned words". *Easy Spell* is a time saver, and in business time is money. C

Creating and Photographing Graphic Displays

by Brooks Cooley

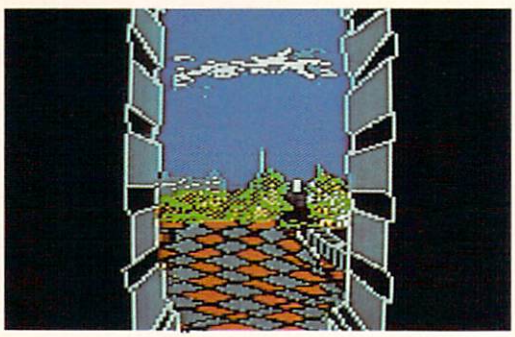
Since the beginning of time man has had the desire to express himself. Early man left sketches and paintings on cave walls that told of life as it was. Today, man still has that innate desire of early ancestors, but over the centuries has developed a highly advanced medium in which to work. Modern advances have led to high-resolution bit-mapped graphics where images are drawn on a graphics pad, digitized, and stored on a disk for later recall. Computers and graphics pads are today what plant dyes and cave walls were to our early ancestors.

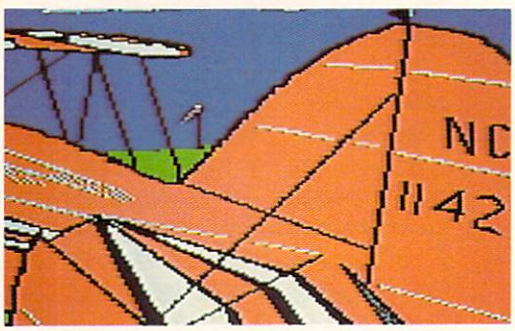
Computer graphics pads have replaced the paper sketch pad of many doodlers as well as serious artists. The ease and speed at which highly detailed displays can be created has astonished many and received the praise of all. No longer is it necessary to spend many hours with pencil and graph paper plotting individual pixel colors. With the aid of a graphics pad, a user can now create in minutes what would have before taken days.

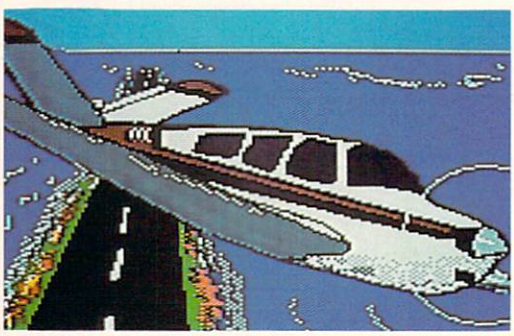
Graphic enhancement peripherals can vary from light pens to touch-sensitive surfaces. Most graphics pads, for example, contain pressure sensitive films that, when touched, send electronic signals to the computer. The pressure of a stylus causes the pad's two voltage-resistant films to come into contact. This determines two distinct voltages, which are sent to an encoding circuit within the graphics pad. An analog-to-digital converter changes these voltages into digital signals. These signals are sent through a control port to, believe it or not, the Commodore 64 SID chip. The SID chip decodes the signals and stores the corresponding values in the appropriate memory locations. The software supporting the graphics pad is now able to determine the X and Y coordinates where the pressure-sensitive film plates came into contact. This process is extremely fast and transparent to the user.

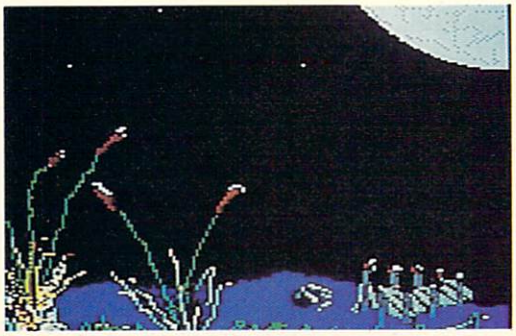
The photographs shown are high resolution displays created on the Commodore 64 with the aid of a graphics pad. Each display was completed in less than one hour! Anyone can easily create computer graphics such as these, and you don't have to be a programmer or an artist. The graphics pad has a vast number of applications bounded only by our imaginations.

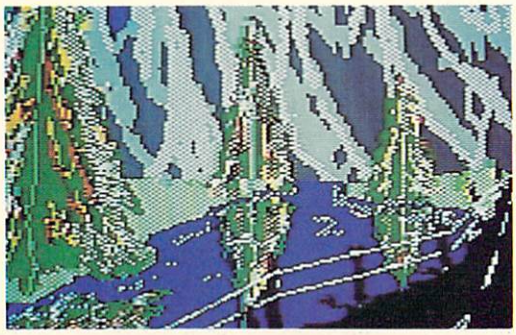
After you have completed a display, you may want to photograph it. You will need a camera that allows you to photograph subjects within 1.5 feet, a tripod, film and a dark room. If a dark room is not available, a hood can be made from any dark, colored material that doesn't let light in. Before taking the photograph, remember to turn off the lights or cover the monitor and camera with the hood in such a way that keeps light from leaking in. This will prevent reflections from showing up in the photograph. If you are using the hood, be careful not to get it in the camera's field of view.

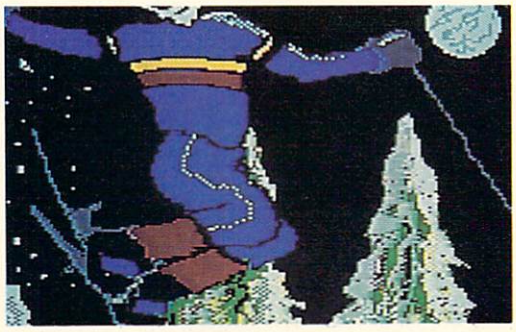












Depending on the camera used, you may have to experiment with different combinations of shutter speeds, screen brightness, and f stops. I have obtained the best results with Kodak Kodacolor II 100 ASA or Kodacolor VR 100 film at f stop 8 with an exposure of one second. A flash should not be used when photographing your displays. If color slides are preferred, substitute Kodak Ektachrome 64 slide film for the print film. When all of the exposures have been made, the film can be taken to any local Kodak film developer for processing.

If you think you have created an exceptional display, please send us a copy of your print or slide. Also include a short description of how you photographed it, your name, address and how long it took you to create the display. Commodore is looking forward to publishing the best photographs received. For more information about graphics pads, contact your local Commodore dealer and start your masterpiece today.

Send your photographs or slides to:

Commodore Business Machines, Inc.,
Computer Graphics Software Division
1200 Wilson Drive
West Chester, PA 19380

All photographs and slides are nonreturnable and become the property of Commodore. C

On the Merits of Touching Up the X-Rays

by Elizabeth Deal

To structure or not to structure is a hot programming question.

A man went to his doctor to find out why he had been suffering terrible pain. After some tests he was told that the X-ray revealed a serious condition that needed surgery. The man asked how much it would cost. "About ten thousand dollars," the doctor replied. The man said, "Doctor, I can't afford that kind of treatment. I am poor and have no health insurance. What can you do for me for fifty dollars?" The doctor thought about this for a few minutes and said: "Well, I could touch up these X-rays."

Structured Programming

A common description of "structured programming" is one that says, "You can indent lines of code and it has no GOTO statements". That's technically correct of course, but is only partly true. I think it steers beginners onto a totally wrong course.

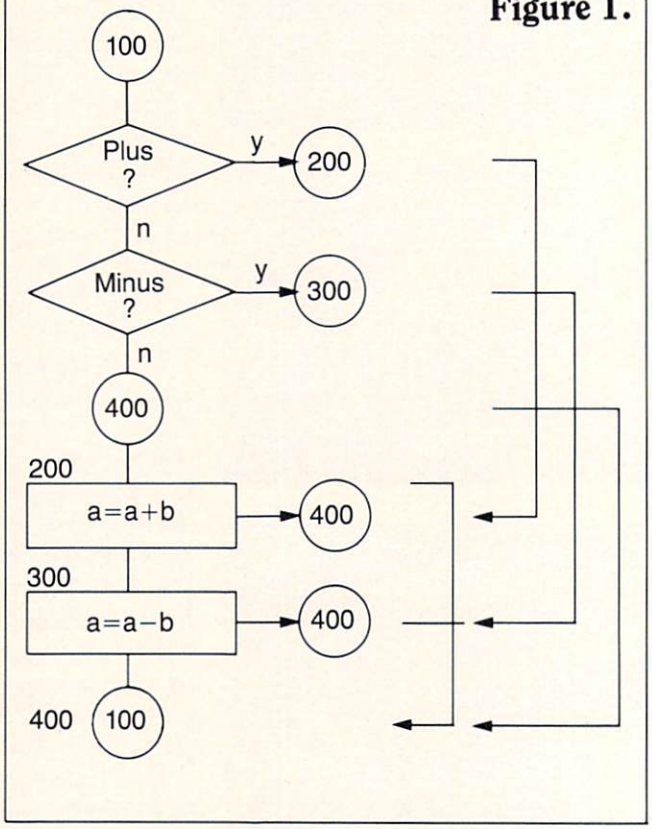
Several months ago I saw just such a description of structured programming in a major computer magazine and I wondered then how much damage it could do. Now I know.

Recently I saw a paper for educators that encouraged them to teach structured programming. For pages and pages it extolled the virtues of structured programming. It described our normal programming efforts as being a "rat's nest of branches, full of illogical thinking" and other such unpardonable sins. Among other things, this paper showed an example of unstructured coding. The program had many GOTOs, with branches crossing all over the place, like this:

```
100...
110 IF PLUS GOTO 200
120 IF MINUS GOTO 300
150 GOTO 400
200 A=A+B: GOTO 400
300 A=A-B: GOTO 400
400 GOTO 100
410...
```

To cure the problem and to demonstrate the power of structuring, the paper proceeded to draw a structured flowchart, whatever that is. Branches that crossed over were neatly replaced by little circles or "termination points" as in Figure 1.

Figure 1.



To complete the job and to compound the confusion, most GOTO statements were carefully replaced by THEN so the program now read:

```
100...
110 IF PLUS THEN 200
120 IF MINUS THEN 300
130 and so on
```

The parting conclusion of this effort was, "We now have a GOTO-less structured program".

In my humble opinion we now have *nothing*.

Replacing GOTO by THEN, a statement that does the same thing only slower, and replacing crossing branches by terminator points is as useful as touching up the X-rays: the mess is still there.

Clearly, this paper missed the point of structuring entirely, which has to do with seeing a large task as a collection of small, perhaps already solved, steps; with desirability of coding tiny units which can be easily debugged; with using flags, the condition of the machine, as a basis for decisions; with using arrays or subroutines in general; with nesting of loops and branches if such must exist, and so on. Instead, the paper played a semantic, substitute-the-keywords, game, never offering simple code such as:

```
100...
110 IF PLUS THEN A=A+B
120 IF MINUS THEN A=A-B
130 GOTO 100
```

which can be cut neater by some Boolean functions anyway. Its uncluttered flowchart could have been the one shown in Figure 2.

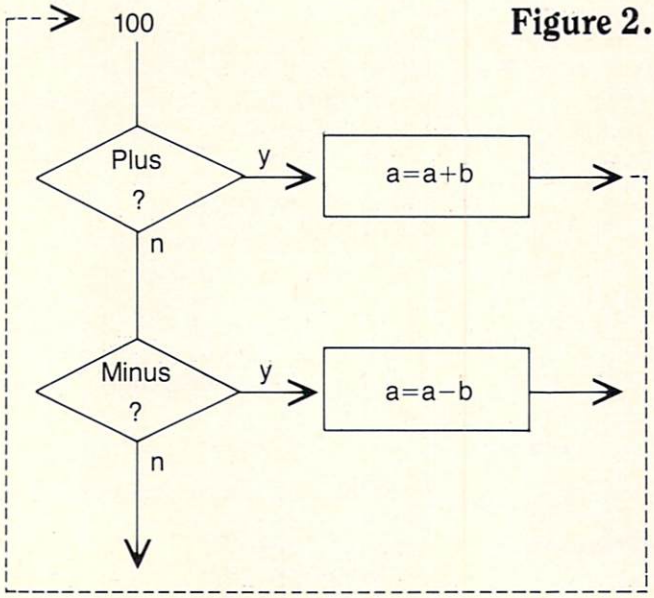


Figure 2.

I do not really intend to take part in or describe the raging debate over structuring, but the concepts of real structured programming are worth knowing indeed. The following are excellent sources, if you're interested in what all the fuss is about:
(1) Various papers by Bohm and Jacopini, Dijkstra and others in the field are available in many libraries.
(2) Paul Nagin and Henry Ledgard, *BASIC With Style: Programming Proverbs,* Hayden Book Company, 1978.
(3) Brian W. Kernighan, P. J. Plauger (Bell Labs, Inc.), *The Elements of Programming Style,* McGraw Hill Book Company, 1974.
(4) Seymour Papert, *Mindstorms: Children, Computers, and Powerful Ideas,* Basic Books, 1980.

BASIC Perspective

BASIC, or machine code for that matter, is difficult to do without branches (GOTOs), but many can be eliminated if we follow this pragmatic principle: code GOSUB when you can, code GOTO if you must.

But keep in mind that this is only a part of the structuring story. While an apparent goal of structuring is reducing unnecessary GOTO statements, the thought behind the exercise is a better program organization for the purpose of readability and easier maintenance in the future. As the code is being reorganized, many GOTOs vanish from sight without any conscious effort. So, in a sense, it sometimes becomes a circular process of one thing helping another, an interesting process indeed.

I often wonder if the structuring fanatics haven't dug their own grave by having overshadowed their sensible campaign for orderly, logical program design with their loud anti-GOTO pronouncements. It leaves the audience with a misleading impression that the entire debate turns on the presence or absence of GOTO and GOTO alone... and we end up with keyword substitution as a cure.

Debugging Troubles

It has crept up slowly, but I began to notice recently that many programmers are abandoning GOTO. I've seen a lot of THEN123 type coding,

and began to wonder why. It's a nasty, inconvenient little switch in style. BASIC-Aid type of utilities, which help search through a program, become difficult to use.

When a program contains GOTOs, and you are interested in "transfer of control" to know if somebody's line 100 is a GOTO/GOSUB target, this type of utility can easily plop all the simple GOTOs on the screen.

But when GOTO100 is coded as THEN100, we're in a terrible predicament of having to weed out genuine THEN-action (as in THEN V=15:K=1) from THEN-line number. Often a program contains so many THEN-action statements that zeroing in on THEN-line numbers becomes an unnecessarily tedious task. Frankly, I fail to see a reason for ever substituting THEN for GOTO. *Please, do not touch up the X-rays!*

I think it's most convenient to code THEN for a process to happen and to code GOTO for transfer to another place in a program. The computer doesn't care, but it can make work much simpler for us. C

Random Thoughts
Part 3: Moment by Moment

by Mark Zimmermann

Last time, we talked about "discrete" distributions of random numbers, that is, distributions that have only a finite, countable number of possible outcomes. In principle you could solve any random number problem as accurately as you pleased using discrete distributions. But in practice it might get very inconvenient and time consuming to do so.

For example, suppose you're studying some simple biological system that has various properties, determined by the sum of the influences of a number of genes. The basic blood type, for instance, might be O, A, B, or AB, depending on only two genes. With so few choices to consider, you can easily work out all the possible cases using discrete probability arguments. But suppose there are 1000 genes that affect some biological property? How can you possibly count up all the possible outcomes?

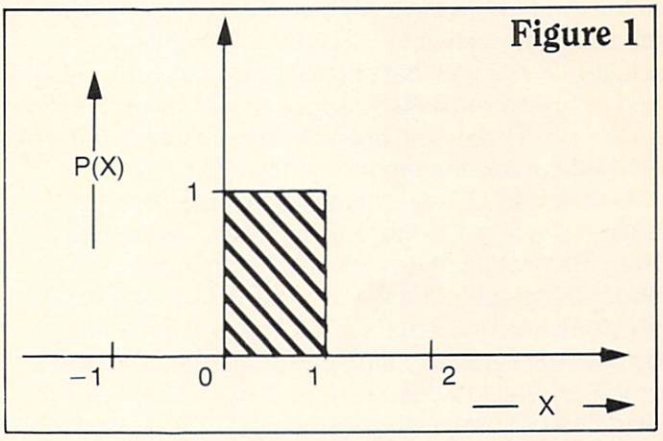
The answer is: don't try! Instead, think about the possible outcomes as a continuous, smoothly varying set of possibilities, and ask how likely it is to come out in any given range of results. To get a precise answer in some complicated cases may require a little advanced mathematics. But most of the time you can go a long way using only a few simple rules, which we'll cover in this and in future columns.

Mind the P's and Q's

To discuss the continuous distributions that we're interested in, we need a way to talk about the quantitative differences between different distributions. A good way to begin is by defining the "moment" of a distribution. If X is the name of the random number we're concerned with and it has some probability distribution we'll call P, then we can define the "Nth moment of the distribution P" as the average value of X↑N.

That definition may not make much sense yet, so let's explore it for a little while. First, for a continuous random variable X, what exactly does a "probability distribution P" mean? Answer: P is a function of X (you can write it as P(X) if you like, and read it as "P of X") which is always positive (or zero) and which describes the chance that X assumes some particular value. For example, look at the graph in Figure 1.

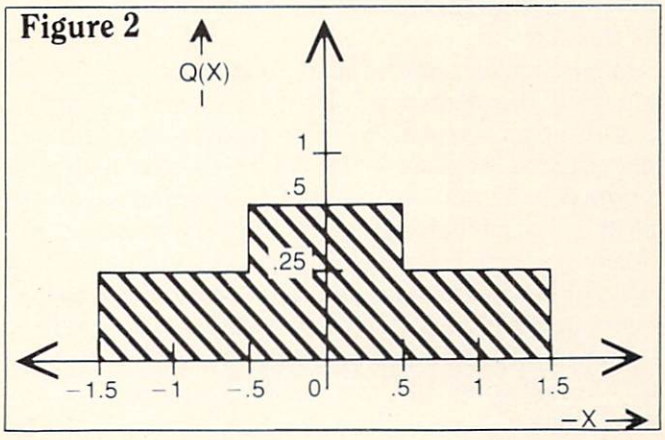
Figure 1



As you can see, we've plotted the value of P along a vertical axis, with X running horizontally. P(X) is zero everywhere except between X=0 and X=1, where P(X)=1. This choice of P(X) just says that X has *zero* probability of being chosen randomly if X is less than zero or X is greater than one. On the other hand, any value of X between zero and one is equally likely to be picked. So the function ("continuous distribution") P(X) graphed in Figure 1 is precisely the RND(1) function of standard BASIC in your computer!

A slightly more complicated probability distribution, called Q(X), is graphed in Figure 2, to help you get the idea even clearer.

Figure 2



As you can see, if Q(X) describes the distribution of the random variable X, then X is equally likely to fall anywhere between −.5 and +.5. However, in addition X has half that chance to fall between −1.5 and −.5, and similarly a chance to fall between +.5 and +1.5. There is no chance (zero probability) for X to fall outside the range −1.5 to +1.5.

Both P and Q are pretty simple probability distributions; we'll get to more complicated examples later. Meanwhile, notice please that the total area under the graph of P (shaded in Figure 1) adds up to precisely one unit squared. It's one unit horizontally by one unit vertically. Similarly, the area under the graph of Q also adds up to one square unit. It's made up of three rectangles of areas .25, .5, and .25 respectively, which adds up to one again. The fact that the areas under the graphs are one is no coincidence. It's a basic fact of probability that X has to take on *some* value, for certain, and a thing that is certain has probability one.

So what I'm trying to say is that the *area* under the graph of a continuous probability distribution adds up to the total chance that the random variable falls into that range along the horizontal axis of the graph. The chance that the variable X when distributed according to probability distribution P(X) falls between zero and one is probability one (certainty), since the area of P's graph between zero and one is one square unit. The chance that X falls between zero and .5 if governed by P, is .5, that is, the area under the graph of P above the X axis between X=0 and X=.5.

In the slightly more complex case of the distribution Q(X), the chance of finding X between −.5 and .5 adds up to .5, the area under the graph of Q for that range of X values. What is the chance of finding X between −1 and zero? Well it's half of the rectangle in Q's graph between −1.5 and −.5, plus half of the rectangle between −.5 and .5—a total of ½*.25+½*.5, which equals .375 (or ⅜). For a final example, the chance of finding X between 1.0 and 1.1 when X is controlled by distribution Q is the area in that zone 0.1* .25 =.025—a pretty small chance.

One Moment, Please

Ok, now let's return to exploring the definition of the "moments" of a distribution. What are some specific cases?

Plugging N=1 into the definition, we read that the "first moment" of a probability distribution is the average value of X↑1, that is, simply the average value of X. Much ado about nothing!? Not at all. The average value of a random variable X is probably one of the first things we should know and keep in mind when thinking about it. For our distribution P(X) from Figure 1, where X is equally likely to fall anywhere between zero and one, the average outcome is exactly .5. For random variables governed by Q in Figure 2, you can guess by symmetry that the average result is zero (since it's equally likely to be positive or negative, and zero is the only number that fits both requirements).

If you study the graphs of the random distributions P and Q, and think about the areas under the graphs as physical objects (such as you'd get by graphing onto thick paper and cutting out the shaded areas), you'll find another meaning of the first moment of a distribution. The first moment corresponds to the center of gravity (or center of mass, if you prefer) of the cutout graph, along the X axis. If you have a complicated distribution and it's not obvious what the average value is, a legitimate way to determine the average (approximately) is to graph the distribution, cut it out and balance it!

Moment Two

On to the second moment: the average value of X squared. What good is that? Well for starters, consider the special case where the first moment was zero (as it was for Q in Figure 2). That first moment came out zero because there was as much plus as there was minus in the numbers adding up to give the average outcome of X. But if we're averaging the values of X↑2, then all of the inputs to the average will be positive, since squaring the negative numbers makes them positive. Note: be sure to square *before* adding up and averaging.

Now squaring a number, as discussed in an earlier column, makes for a funny non-linear stretching of

values. Numbers near zero get smaller when squared, but numbers greater than one get bigger. So the second moment, the average of the squares of the random outcomes, is a "weighted average" that gives more weight to bigger numbers than it does to numbers near zero. In fact, the second moment tells you a lot about *how wide* a random distribution is: the wider the distribution, the bigger the second moment tends to be.

The second moment, in fact, is used to define the "variance" and the "standard deviation" of a random variable about its average value. The variance, call it V, is just the difference between the second moment and the square of the first moment. The standard deviation, call it S, is the square root of the variance. Just looking at the dimensions of the things, the definitions make sense. The first moment, the average value of X, has the same units as X does. The second moment has units of X squared. So, subtracting the square of the first moment from the second is dimensionally consistent; you aren't trying to subtract apples from oranges. Taking the square root of the variance gets you back to units of X for the standard deviation.

Physically, the second moment of a distribution corresponds to the *moment of inertia* of the cut-out graph of the distribution about the vertical axis. That makes sense. A wide, spread-out probability distribution has a large second moment, just as a skater with arms stretched out has a large moment of inertia and spins slower. The terminology "moments of a distribution", in fact, probably arises from this physical meaning of the second moment.

The Monte Carlo Method

To solidify the concept, let's determine the second moment of our distributions P and Q. This is not as trivial an enterprise as finding the first moments, which we could do by inspection. In fact, to get the exact answer without a lot of trickery one probably has to use a little calculus. I'll give you that exact answer, but first we should explore a couple of methods that you can use without calculus to get as accurate a result as you please.

The first method, which has great generality, is usually called the "Monte Carlo" approach, named after the famous casino. In a Monte Carlo calculation, you simulate a random process by doing a numerical experiment on your computer over and over again until the average result settles down to whatever accuracy you like. Monte Carlo calculations are very popular in complicated situations where an exact computation is too horrendous to think about doing.

Start out with the random distribution P(X), which is simply RND(1) on your computer. To get the second moment, the average value of the random variable X↑2, you might try inputting:

```
A=0: FOR I=1 TO 100: X=RND(1)
:A=A+X*X: NEXT I: PRINT A/100.
```

In a short time, you should see the resulting estimate of the second moment appear on your display. When I first tried it, I got 0.3276; my second try gave 0.3086. If you have the patience, change the number "100" in the one-liner above to a larger value, and you'll get a more accurate average outcome.

Slicing it Fine

If putting yourself at the mercy of a random number generator makes you uncomfortable, here's another approach to computing the second moment of a distribution. Split the distribution up into a bunch of thin vertical slices, estimate the average value of X squared in each slice and add up those estimates times the probability of finding X in each slice. That is, estimate how often X shows up in each slice you cut and average the estimates of X squared weighted by the chance of each occurrence. For example, look at the graph of P in Figure 1. Cut the X axis up into five slices between X=0 and X=1, with cuts at X=.2,.4,.6, and .8. The chance of finding X in each of those slices is the area under the graph of P, namely 1/5 =0.2. In the first slice, where X ranges between zero and 0.2, the midpoint value is 0.1, and so a reasonable estimate for the average value of X↑2 is 0.1↑2 = 0.01. Similarly, if you pick the midpoint of each thin slice and square it to estimate the average of X squared in that slice, you come up with the five estimates for the five slices:

0.01, 0.09, 0.25, 0.49, and 0.81. Multiplying each number by the chance of finding X in that slice and adding up the total, you get the weighted average of 0.33 as your estimate for the second moment of the distribution P.

You could automate this procedure and write a program to do the estimating and adding for you. In fact, you could also work out mathematically the general form of the sum as a function of the number of slices you cut the interval into. If you applied your program (or your analysis of the math) to the case where the number of slices was huge, you'd find that the answer you got for the second moment approached the number 1/3. And, in fact, 1/3 is exactly the answer of the problem. The usual freshman-calculus solution follows this method to the limit where the number of thin slices goes to infinity. But you see that even with as few as five slices, we got the right result to better than two decimal places!

If you want to test yourself by working out the second moment of random distribution Q in Figure 2, you can check your results against the exact answer at the end of this article.

Other Moments

How about the variance and the standard deviation of distribution P? Well, the variance is the second moment (1/3) minus the square of the first moment (1/2 squared = 1/4) or V=1/12. The standard deviation is SQR(V) = 1/SQR(12) = .288675. Roughly speaking, the standard deviation tells you the spread, plus or minus, of a "typical" choice of random numbers about the average value.

What about higher moments of a distribution? They tend to come up less often than the first (average) and second (variance) moments in practical applications but they're still important. The third moment, the average of X cubed, tells you something about the *asymmetry* of the random distribution, the bias about the center toward plus or minus. The fourth moment gives more information about how steeply the distribution falls off at the edges—how rounded it is. And so on.... It's a surprising and marvelous mathematical fact that given all the moments of a distribution, you can deduce what the distribution looks like. This ability to go backwards from the moments to the graph of the distribution is similar to the way you can analyze any sound into the frequency components that go into it, or analyze a color into the wavelengths of light that made it (using a prism, for example).

One more interesting item to think about: what is the *zeroth* moment of a distribution? By our definition, it is the average value of X↑0, and any number to the zero power is one. The zeroth moment corresponds to the total area under the probability distribution graph and, as we discussed earlier, that area has to add up to one square unit. So, even the zeroth moment has a sensible interpretation.

The "Square Root" Rule of Thumb

In closing this time let me give you a very useful, approximate rule that helps with random numbers in many situations. I call it the "square root" rule. On the average, the deviation of the sum of a bunch of random numbers goes up proportionally with the square root of the number of numbers added.

So for instance, if you toss a hundred pennies at once, how far from the average value of 50 should you expect the outcome to be? Answer: roughly plus or minus ten, the square root of 100. If you tossed a million pennies, you'd expect the outcome to be 500,000 plus or minus 1,000 since 1,000 is the square root of 1,000,000.

If you do a Monte Carlo experiment for 100 trials (as we did in estimating the second moment of P earlier) and the results of a single trial vary by ±.5 or so, you'd expect the sum of the hundred trials to vary by ten times that, ±5, and so the average result should vary by ±5/100 = ±.05. Increasing the number of trials by a factor of 100 should give you one more decimal digit of accuracy in your answer.

This "square root" rule tells you that a political poll of 1000 people is likely to get fluctuations in the responses by ±30 or so, that is, roughly ±3% accuracy. The rule applies to lots of situations.

Meanwhile, the second moment of distribution Q, for those of you who want to know, turns out to be 7/12 = .583333.... C

Using Picture Format
A Six Pack +

by F. H. Shedd

The first statement that is sadly missed by the programmer who graduates from one of the higher level languages to the BASIC used by most microcomputers is the lack of a PRINT formatting provision. COBOL has the PICTURE, FORTRAN the FORMAT and others the USING statement. The well detailed programmer's reference guides published by Commodore give information to the gamester who may wish to put a tear in Ms Pacman's eye, but gives little aid to the business programmer trying to align his decimal points when generating an accounting report.

The reader will probably have already deduced that I am an old retired systems analyst who had been accustomed to satisfying accountants who would sometimes look at a printout and imply that it had all the esthetic value of a ransom note. The greatest opportunity for employment in the computing science field lies in business programming. Working there a business programmer who does not visibly accept the fact that computing science is the servant of financial management, not its master, will soon be set right during his first meeting with the comptroller, who in many companies is over the manager of the computing organization. In computerese: if unconvinced then unemployed.

And it might also pop into the reader's mind that I had been accustomed to having access to million-dollar mainframes and that at this time I have a Commodore 64 to keep me happy.

Now, even though I do not have to satisfy others, I cannot put up with sloppy outputs. They are the mark of laziness, ignorance or the professional amateur. Of these ignorance is most easily forgiven for it is the most easily remedied. So one of the first subroutines that I developed, after having solved some of the mysteries of the LEFT, MID and RIGHT handling of a string was to write a short program which furnishes the four most needed formats. It occupies a minimum of RAM and, when compared to some other programs which do less and use more memory, is a miracle of simplicity.

Following is a description of the 7 PICTURES pictures made available by the SUBROUTINE "V0123456$" shown as Listing 1. The lower case "s" when appearing in the PICTURE denotes the space reserved for the signum. It is blank for zero and all numbers greater than zero. It contains a dash ("-") for all numbers less than zero. The lower case "b" denotes a blank space.

Instructions 2000 through 2070 allow the user to change output length to satisfy the expected magnitude of input or to fill the space allotted by form design. Do not be too frugal. Remember that input values might be small but their total much greater and the decimal point must still stay in alignment.

Numbers with over eight significant digits are to be avoided when using the 64 for accounting records which must be accurate to the last printed digit. The ninth digit, especially when the product of calculations where repeating decimals are involved, are unreliable. If you like to test this take the square root of 999 and square the result. Note that the value is off three digits in the ninth place. This is the result of the CPU's automatic rounding. When working with numbers such as those that define the distance to the moon, or the National Debt, one requires a "bit" more precision than possessed by the 64.

Although I have never met an accountant who liked to work in cents, if you do have such input just add to the program:

```
2001 V=V/100
```

One last warning about presenting financial reports to an accountant for approval. Never, even as a joke, give him one showing any value in scientific notation. I feel most fortunate that I was at a distance of 20 feet when I witnessed a tragedy of this nature.

If you intend to store these formats in an array and have to be able to restore the V rounded value for future computations you may easily do so using the VAL numeric function for V0$, V1$, and V2$. The $ sign may even be stripped from V3$ and then evaluated. However the forms with embedded commas are much more difficult and it is recommended to double-store the value in your array with one in a simple form.

V0$ — PICTURE s99999999b
- — Length is 10, left justified.
- — Value is identical to V input.
- — Usual use is to record actual (unrounded) V values in an array where constant length record fields are required. Not usually an output FORMAT.

V1$ — PICTURE bs99999999
- — Value is V rounded in cents.
- — Length is 10, right justified.
- — Column heading required : VALUE IN CENTS :.
- — Although accountants dislike financial reports having integer cents as output, this FORMAT is sometimes a godsend to the programmer laying out a Financial Spread Sheet with a Title/Identity column, 12 Monthly columns, and a Total Column. With the limit of a 132-character line these 13 extra spaces can come in mighty handy.
- — This is the best FORMAT to save in an array where V may be recovered for later computations by VAL(V1$)/100.

V2$ — PICTURE bs999999.99
- — Length is 11, right justified.
- — Value is V rounded to nearest cent in decimal dollars.
- — Column heading required : VALUE $:.
- — V may be recovered using VAL (V2$)

V3$ — PICTURE b$s999999.99
- — Length is 12, right justified.
- — Column heading required : VALUE :
- — Incorporates floating $ sign.
- — V may be recovered by using:
  L=LEN(V3$) :V=VAL (RIGHT$(V3$,L-1))

V4$ — PICTURE b(999,999.99)
- — Length is 13, right justified.
- — Incorporates:
  Embedded commas.
  Signum removed. Negative numbers now with leading and trailing brackets.
  Positive numbers and 0 have leading and trailing blanks.
- — This FORMAT is much favored by accountants. Most dislike dash "—" as a CREDIT indicator.
- — It is generally the only approved FORMAT for use when generating Balance Sheets or Profit and Loss Statements.

V5$ — PICTURE b999,999.99 CR
- — Length 14, right justified.
- — Incorporates:
  Embedded commas.
  Negative numbers are suffixed by 'bCR'.
  Positive and 0 by 'bbb' (3 blanks).
- — This FORMAT and V6$ following are invaluable for Billing or Invoicing Forms where space is not at a premium and the printed value must not be subject to misunderstanding.

V6$ — PICTURE b$999,999.99 CR
- — Length 15, right justified.
- — Same as V5$ with added floating $ sign. **C**

Listing 1. Subroutine V0123456$

```
2000 REM** SUBROUTINE  ''V0123456$''
2010 L0=10:REM V0$   JUSTIFIED V
2020 L1=10:REM V1$ -> 12345678<- ROUNDED
2030 L2=11:REM V2$ -> 123456.78<- ROUNDED
2040 L3=12:REM V3$ ->$ 123456.78<- ROUNDED
```

```
2050 L4=13:REM V4$ -)(123,456.78)(- ROUNDED
2060 L5=14:REM V5$ -)  123,456.78 CR(-
2070 L6=15:REM V6$ -)$123,456.78 CR(-
2080 V0$=STR$(V)
2090 L=LEN(V0$):IF L(L0THEN V0$=V0$+" ":GOTO2090
2100 VR=V*100+.5:VY=VR:  VR =(INT(VR))/100:V$=STR$(VR)
2110 VX=VAL(V$)*100:V1$=STR$(VX)
2120 L=LEN(V1$):IF L(L1 THEN V1$=" "+V1$:GOTO2120
2130 Q=LEN(V$)
2140 FOR L=Q TO 1 STEP -1
2150 IF MID$(V$,L,1)()"."THEN NEXT L:V$=V$+".00":GOTO2170
2160 IF L=Q-1 THEN V$=V$+"0"
2170 V2$=V$:V3$="$"+V$
2180 L=LEN(V2$):IFL(L2 THEN V2$=" "+V2$:GOTO2180
2190 L=LEN(V3$):IFL(L3 THEN V3$=" "+V3$:GOTO2190
2200 NG=0:IF V(0THEN NG=1
2210 L=LEN(V$):L=L-1:V4$=RIGHT$(V$,L)
2220 V5$=V4$:IF L=(6 THEN GOTO2260
2230 VA$=LEFT$(V4$,L-6):VB$=RIGHT$(V4$,6):V4$=VA$+","+VB$:V5$=V4
$
2240 V5$=V4$:IF L =(10 THEN GOTO2260
2250 L=L+1:VA$=LEFT$(V4$, L-10):VB$=RIGHT$(V4$,10):V4$=VA$+","+V
B$
2260 IF NG THEN V4$="("+V4$+")":GOTO2280
2270 V4$=" "+V4$+" "
2280 L=LEN(V4$):IF L(L4 THEN V4$=" "+V4$:GOTO2280
2290 IF NG THEN V5$=V5$+" CR":GOTO2310
2300 V5$=V5$+"    "
2310 V6$="$"+V5$
2320 L=LEN(V5$):IF L(L5 THEN V5$=" "+V5$:GOTO2320
2330 L=LEN(V6$):IF L(L6 THEN V6$=" "+V6$:GOTO2330
2340 RETURN
```

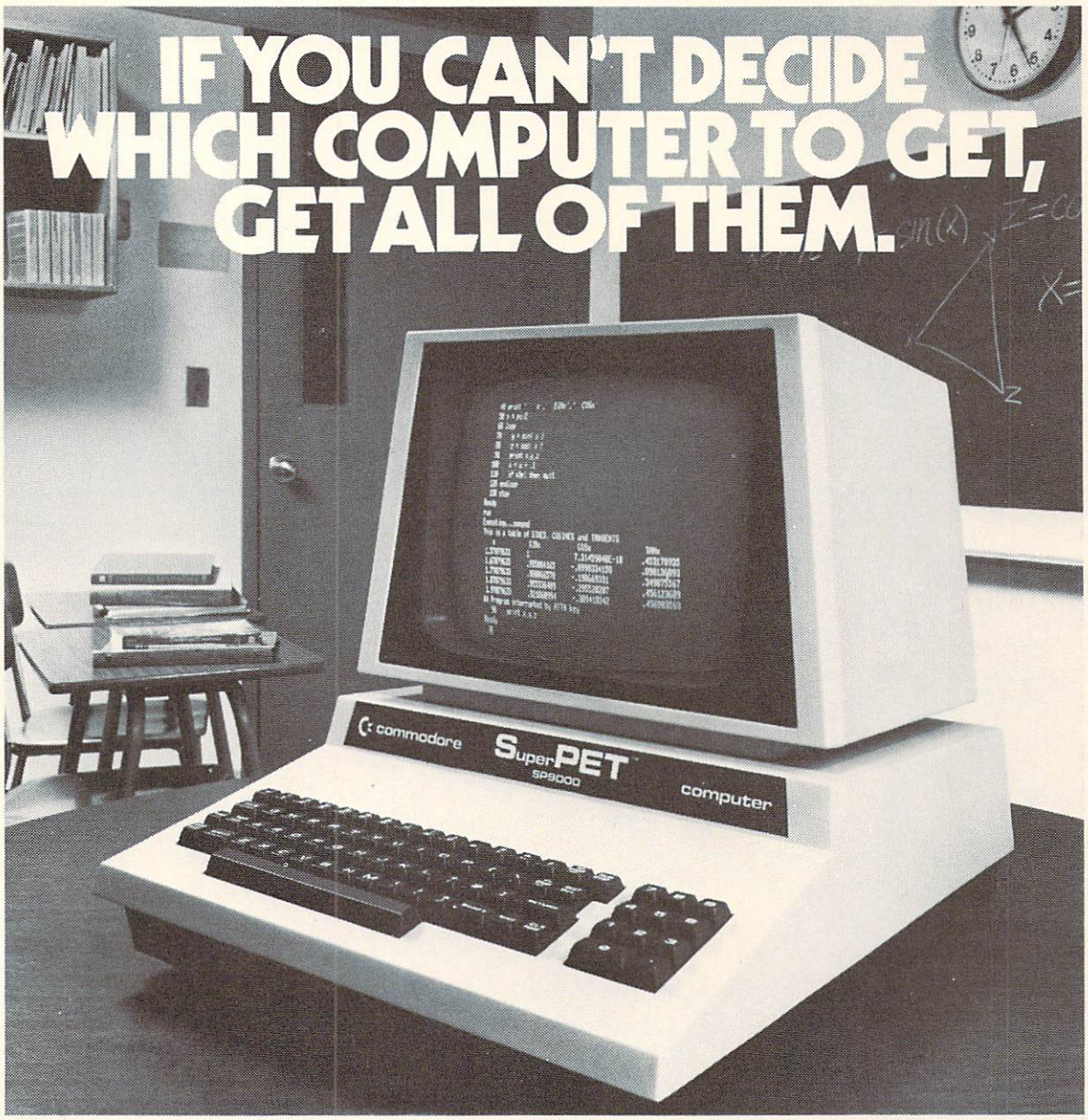
Listing 2. Demo Screen Dump

```
     Note: Printer need not be on line to run.
           See Instuction 160.
100 REM  ''DEMO SCREEN DUMP''
105 REM DUMP FOR OKIDATA 82A
110 REM  F H SHEDD
120 REM  FAIR OAKS, CA   95628
130 REM      916-961-1192
```

```
140 CL$=CHR$(147):PRINT CL$
150 PRINT"1 = PRINTER ON LINE FOR SCREEN DUMP"
160 PRINT"0 = NO PRINTER"
170 INPUT P1
180 OP=0
190 SU=0:SR=0:OP=OP+1:IF OP=7 THEN STOP
200 PRINT CL$;
210 IFOP (5 THEN PRINT              "V = ENTRY    * VR  ROUNDED* ";
212 IF OP)4 THEN PRINT " V ENTERED    *  ";
220 IF OP=1 THEN PRINT" V1$ FORMAT"
230 IF OP=1 THEN PRINT"              *             * VALUE=CENTS"
240 IF OP=2 THEN PRINT " V2$ FORMAT"
250 IF OP=2 THEN PRINT "             *             *   $ VALUE"
260 IF OP=3 THEN PRINT " V3$ FORMAT"
270 IF OP=4 THEN PRINT "   V4$ FORMAT"
280 IF OP=4 THEN PRINT "             *
282 IF OP=5 THEN PRINT "   V5$ FORMAT"
284 IF OP=5 THEN PRINT "                          $ VALUE"
286 IF OP=6 THEN PRINT " V6$ FORMAT"
290 PRINT "======================================="
300 READ V
310 VT$=STR$(V)
320 L=LEN(VT$):IF L(10 THEN VT$=VT$+" ":GOTO 320
330 IF V)1E+10 THEN GOTO 420
340 GOSUB 2000
350 IF OP(5 THEN PRINT VT$;"  *";V0$;" *";
355 IF OP)4 THEN PRINT VT$;"   * ";
360 IF OP=1 THEN PRINT V1$
370 IF OP=2 THEN PRINT V2$
380 IF OP=3 THEN PRINT V3$
390 IF OP=4 THEN PRINT V4$
392 IF OP=5 THEN PRINT V5$
394 IF OP=6 THEN PRINT V6$
400 SU = SU + V : SR =SR+VR
410 GOTO 300
420 RESTORE
430       V=SR:GOSUB 2000
440 PRINT "SUM V = ";SU:PRINT  "  SUM V ROUNDED = ";V6$
450 PRINT "["
460 GOSUB 510
465                          STOP
470 IF P1 THEN GOSUB 7000
480 GOTO 190
```

```
490 DATA .089,-1.1,0,123.456,-1234.552,123456.78,-9,23.499,.1,10
0,-.999
500  DATA 1E+11
510 PRINT"      PRESS SPACE BAR TO CONTINUE"
520 GET W$:IF W$=""THEN GOTO 520
530 RETURN
2000 REM** SUB PROGRAM ''V0123456$''
2010 L0=10:REM V0$    JUSTIFIED V
2020 L1=10:REM V1$ -> 12345678<- ROUNDED
2030 L2=11:REM V2$ -> 123456.78<- ROUNDED
2040 L3=12:REM V3$ ->$ 123456.78<- ROUNDED
2050 L4=13:REM V4$ ->(123,456.78)<- ROUNDED       OR
2060 L5=14:REM V5$ -> 123,456,78 CR<-
2070 L6=15:REM V6$ ->$123,456.78 CR<-
2080 L=LEN(V0$):IF L<L0THEN V0$=" "+V0$:GOTO2080
2090 VR=V*100+.5:VY=VR:  VR =(INT(VR))/100:V$=STR$(VR)
2100 VX=VAL(V$)*100:V1$=STR$(VX)
2110 L=LEN(V1$):IF L<L1 THEN V1$=" "+V1$:GOTO2110
2120 Q=LEN(V$)
2130 FOR L=Q TO 1 STEP -1
2140 IF MID$(V$,L,1)<>"."THEN NEXT L:V$=V$+".00":GOTO2160
2150 IF L=Q-1 THEN V$=V$+"0"
2160 V2$=V$:V3$="$"+V$
2170 L=LEN(V2$):IFL<L2 THEN V2$=" "+V2$:GOTO2170
2180 V0$=V2$
2190 L=LEN(V3$):IFL<L3 THEN V3$=" "+V3$:GOTO2190
2200 NG=0:IF V<0THEN NG=1
2210 L=LEN(V$):L=L-1:V4$=RIGHT$(V$,L)
2220 V5$=V4$:IF L=<6 THEN GOTO2260
2230 VA$=LEFT$(V4$,L-6):VB$=RIGHT$(V4$,6):V4$=VA$+","+VB$:V5$=V4
$
2240 V5$=V4$:IF L =<10 THEN GOTO2260
2250 L=L+1:VA$=LEFT$(V4$, L-10):VB$=RIGHT$(V4$,10):V4$=VA$+","+V
B$
2260 IF NG THEN V4$="("+V4$+")":GOTO2280
2270 V4$=" "+V4$+" "
2280 L=LEN(V4$):IF L<L4 THEN V4$=" "+V4$:GOTO2280
2290 IF NG THEN V5$=V5$+" CR":GOTO2310
2300 V5$=V5$+"   "
2310 V6$="$"+V5$
2320 L=LEN(V5$):IF L<L5 THEN V5$=" "+V5$:GOTO2320
2330 L=LEN(V6$):IF L<L6 THEN V6$=" "+V6$:GOTO2330
2340 RETURN
7000 REM  SCREEN DUMP - F H SHEDD 9/83
```

```
7005 REM DUMP FOR OKIDATA 82A
7010 REM LEFT BRACKET [ ENDS DUMP
7020 SI$=CHR$(15):BS$=CHR$(8):PO$=CHR$(16):RV$=CHR$(18):RO$=CHR$
(146)
7030 QT$=CHR$(34):MF$=CHR$(145):VR=PEEK(648)*256
7040 OPEN 4,4,10
7050 PRINT#4,"..............................................."
7060 PRINT#4,":         S-C-R-E-E-N  D-U-M-P                :"
7070 PRINT#4,"***********************************************"
7080 FOR CL = 0 TO 24
7090  QF=0:AS$=MF$
7100    FOR RO = 0 TO 39
7110 SC= PEEK(VR+40*CL+RO)
7120 IF SC = 34 THEN QF = 1- QF
7130 IF SC<>162 THEN7160
7140 QF= 1 -QF: IF QF = 1 THEN AS$=RV$+QT$:GOTO 7220
7150 AS$= AS$+QT$+RO$:GOTO 7180
7160 IF QF = 1 AND (SC>128) THEN SC = SC -128:GOTO 7180
7170 IF SC>= 128 THEN SC = SC - 128:RF=1:AS$=AS$+RV$
7180 IF SC< 32 OR SC > 95 THEN AS = SC+ 64:GOTO 7210
7190 IF SC >31 AND SC < 64 THEN AS = SC:GOTO 7210
7200 IF SC>63 AND SC<96 THEN AS = SC+32:GOTO 7210
7210 AS$=AS$+CHR$(AS)
7220 IF RF = 1 THEN A$=AS$+RV$:RF=0
7230 IF AS =91 THEN GOTO 7290
7240     NEXT RO
7250 IF QF = 0 THEN PRINT#4,SI$PO$"* "AS$"*":GOTO 7270
7260 PRINT#4,SI$+PO$+"20"+AS$+QT$
7270    NEXT CL
7280 PRINT#4,SI$
7290 PRINT#4,"***********************************************"
7300 CLOSE 4
7310 RETURN
```



THE SUPERPET.™ 5 LANGUAGES

It's not easy finding a microcomputer that can fulfill all of your requirements. Unless you've discovered the Commodore SuperPET.™

The SuperPET is the *only* microcomputer that comes with five of the most powerful structured languages: MicroBASIC, MicroAPL, MicroFORTRAN, MicroPASCAL, and MicroCOBOL.

We think you'll agree it's far and away the most complete computer you can buy for your school.

commodore COMPUTER

Screen Dumps of Demonstration Data: Program "Demo Screen Dump"

```
:          S-C-R-E-E-N   D-U-M-P                :
*****************************************
* V = ENTRY    * VR   ROUNDED*  V1$ FORMAT  *
*              *              * VALUE=CENTS *
* ==================================== *
*  .089        *        .09 *          9   *
* -1.1         *      -1.10 *       -110   *
*  0           *       0.00 *          0   *
*  123.456     *     123.46 *      12346   *
* -1234.552    *   -1234.55 *    -123455   *
*  123456.78   *  123456.78 *   12345678   *
* -9           *      -9.00 *       -900   *
*  23.499      *      23.50 *       2350   *
*  .1          *        .10 *         10   *
*  100         *     100.00 *      10000   *
* -.999        *      -1.00 *       -100   *
* SUM V =   122458.273                     *
*   SUM V ROUNDED =  $122,458.28           *
*****************************************
```

```
:          S-C-R-E-E-N   D-U-M-P                :
*****************************************
* V = ENTRY    * VR   ROUNDED*  V3$ FORMAT  *
* ==================================== *
*  .089        *        .09 *       $ .09  *
* -1.1         *      -1.10 *      $-1.10  *
*  0           *       0.00 *      $ 0.00  *
*  123.456     *     123.46 *    $ 123.46  *
* -1234.552    *   -1234.55 *   $-1234.55  *
*  123456.78   *  123456.78 * $ 123456.78  *
* -9           *      -9.00 *      $-9.00  *
*  23.499      *      23.50 *    $ 23.50   *
*  .1          *        .10 *      $ .10   *
*  100         *     100.00 *    $ 100.00  *
* -.999        *      -1.00 *      $-1.00  *
* SUM V =   122458.273                     *
*   SUM V ROUNDED =  $122,458.28           *
*****************************************
```

```
:          S-C-R-E-E-N   D-U-M-P                :
*****************************************
* V = ENTRY    * VR   ROUNDED*  V2$ FORMAT  *
*              *              *   $ VALUE   *
* ==================================== *
*  .089        *        .09 *        .09   *
* -1.1         *      -1.10 *      -1.10   *
*  0           *       0.00 *       0.00   *
*  123.456     *     123.46 *     123.46   *
* -1234.552    *   -1234.55 *   -1234.55   *
*  123456.78   *  123456.78 *  123456.78   *
* -9           *      -9.00 *      -9.00   *
*  23.499      *      23.50 *      23.50   *
*  .1          *        .10 *        .10   *
*  100         *     100.00 *     100.00   *
* -.999        *      -1.00 *      -1.00   *
* SUM V =   122458.273                     *
*   SUM V ROUNDED =  $122,458.28           *
*****************************************
```

```
:          S-C-R-E-E-N   D-U-M-P                :
*****************************************
* V = ENTRY    * VR   ROUNDED*    V4$ FORMAT*
*              *              *              *
* ==================================== *
*  .089        *        .09 *          .09  *
* -1.1         *      -1.10 *       (1.10)  *
*  0           *       0.00 *         0.00  *
*  123.456     *     123.46 *       123.46  *
* -1234.552    *   -1234.55 *   (1,234.55)  *
*  123456.78   *  123456.78 *  123,456.78   *
* -9           *      -9.00 *        (9.00)  *
*  23.499      *      23.50 *        23.50   *
*  .1          *        .10 *          .10   *
*  100         *     100.00 *       100.00  *
* -.999        *      -1.00 *        (1.00)  *
* SUM V =   122458.273                     *
*   SUM V ROUNDED =  $122,458.28           *
*****************************************
```

```
.....................................
:        S-C-R-E-E-N  D-U-M-P        :
*************************************
*  V ENTERED   *     V5$ FORMAT      *
*              *       $ VALUE       *
* ============ * ===================  *
*  .089        *          .09        *
* -1.1         *         1.10 CR     *
*  0           *         0.00        *
*  123.456     *       123.46        *
* -1234.552    *    1,234.55 CR      *
*  123456.78   *  123,456.78         *
* -9           *         9.00 CR     *
*  23.499      *        23.50        *
*  .1          *          .10        *
*  100         *       100.00        *
* -.999        *         1.00 CR     *
* SUM V =  122458.273                *
*   SUM V ROUNDED =  $122,458.28     *
*************************************
```

```
.....................................
:        S-C-R-E-E-N  D-U-M-P        :
*************************************
*  V ENTERED   *    V6$ FORMAT       *
* ============ * ===================  *
*  .089        *          $.09       *
* -1.1         *         $1.10 CR    *
*  0           *         $0.00       *
*  123.456     *       $123.46       *
* -1234.552    *    $1,234.55 CR     *
*  123456.78   *  $123,456.78        *
* -9           *         $9.00 CR    *
*  23.499      *        $23.50       *
*  .1          *          $.10       *
*  100         *       $100.00       *
* -.999        *         $1.00 CR    *
* SUM V =  122458.273                *
*   SUM V ROUNDED =  $122,458.28     *
*************************************
```

Listing 3. Demo Printer

```
100 REM ''DEMO PRINTER''  OKIDATA 82A
110 OPEN 1,4,15
120 PRINT#1,CHR$(27)+CHR$(66):REM OKI SHORT LINE
130 PRINT CHR$(147):PRINT#1,CHR$(29):REM OKI CONDENSED
140 L1$="============================="
150 H1$="V0$ LEN 10    :V1$ LEN 10  :"
160 H5$=" INPUT V      : ROUNDED    :"
170 A1$=" ACTUAL       :  CENTS     :"
180 L2$="============================="
190 H2$="V2$  LEN 11    :V3$   LEN12  :"
200 H6$="  VALUE         :             :"
210 A2$="    $           :  VALUE      :"
220 L3$= "================================="
230 H3$="V4$     LEN 13 :V5$      LEN 14 :"
240 H8$="     VALUE      :       VALUE    :"
250 A3$="       $        :         $      :"
260 L4$="==============="
270 H4$="V6$      LEN 15 "
280 H9$="     VALUE"
290 A4$=""
300 PRINT#1,L1$+L2$+L3$+L4$
310 PRINT#1,H1$+H2$+H3$+H4$
320 PRINT#1,H5$+H6$+H7$+H8$+H9$
330 PRINT#1,A1$+A2$+A3$+A4$
340 PRINT#1,L1$+L2$+L3$+L4$
345 SU=0:SR=0
350 READ V
355 IF V > 999999999 THEN GOTO 500
360 GOSUB 2000
400 PRINT#1, V0$;"    *";V1$;" * ";V2$;"    *";V3$;" *";V4$;" *";V
5$;" *";V6$
410 SU =SU+V:SR=SR+VR
450 GOTO 350
500 RESTORE
510 PRINT#1,L1$+L2$+L3$+L4$
520 PRINT#1," UNROUNDED TOTAL = ";SU
530 V=SR:GOSUB2000
540 PRINT#1,"   ROUNDED TOTAL = ";V6$
900 CLOSE 1:STOP
1000 DATA 12345,123456.78,-1.236
1010 DATA -12341.763,45.469,0,-10.111,122.987,-555.5211,.11678,9
09.995
1100 DATA 1E+10
```

```
2000 REM** SUB PROGRAM ''V0123456$''
2010 L0=10:REM V0$    JUSTIFIED V
2020 L1=10:REM V1$ -) 12345678(- ROUNDED
2030 L2=11:REM V2$ -) 123456.78(- ROUNDED
2040 L3=12:REM V3$ -)$ 123456.78(- ROUNDED
2050 L4=13:REM V4$ -)(123,456.78)(- ROUNDED       OR
2060 L5=14:REM V5$ -) 123,456.78 CR(-
2070 L6=15:REM V6$ -)$123,456.78 CR(-
2075 V0$=STR$(V)
2080 L=LEN(V0$):IF L(L0THEN V0$=V0$+" ":GOTO2080
2090 VR=V*100+.5:VY=VR:  VR =(INT(VR))/100:V$=STR$(VR)
2100 VX=VAL(V$)*100:V1$=STR$(VX)
2110 L=LEN(V1$):IF L(L1 THEN V1$=" "+V1$:GOTO2110
2120 Q=LEN(V$)
2130 FOR L=Q TO 1 STEP -1
2140 IF MID$(V$,L,1)(>"."THEN NEXT L:V$=V$+".00":GOTO2160
2150 IF L=Q-1 THEN V$=V$+"0"
2160 V2$=V$:V3$="$"+V$
2170 L=LEN(V2$):IFL(L2 THEN V2$=" "+V2$:GOTO2170
2180 REM  V0$=V2$
2190 L=LEN(V3$):IFL(L3 THEN V3$=" "+V3$:GOTO2190
2200 NG=0:IF V(0THEN NG=1
2210 L=LEN(V$):L=L-1:V4$=RIGHT$(V$,L)
2220 V5$=V4$:IF L=(6 THEN GOTO2260
2230 VA$=LEFT$(V4$,L-6):VB$=RIGHT$(V4$,6):V4$=VA$+","+VB$:V5$=V4
$
2240 V5$=V4$:IF L =(10 THEN GOTO2260
2250 L=L+1:VA$=LEFT$(V4$, L-10):VB$=RIGHT$(V4$,10):V4$=VA$+","+V
B$
2260 IF NG THEN V4$="("+V4$+")":GOTO2280
2270 V4$=" "+V4$+" "
2280 L=LEN(V4$):IF L(L4 THEN V4$=" "+V4$:GOTO2280
2290 IF NG THEN V5$=V5$+" CR":GOTO2310
2300 V5$=V5$+"   "
2310 V6$="$"+V5$
2320 L=LEN(V5$):IF L(L5 THEN V5$=" "+V5$:GOTO2320
2330 L=LEN(V6$):IF L(L6 THEN V6$=" "+V6$:GOTO2330
2340 RETURN
2350 END
```

programmer's tips

Demonstration of Printed Output Using Subroutine V0123456$

```
V0$ LEN 10   :V1$ LEN 10  :V2$  LEN 11   :V3$   LEN12  :V4$    LEN 13 :V5$     LEN 14 :V6$     LEN 15
 INPUT V     : ROUNDED    :  VALUE      :              :    VALUE      :     VALUE      :    VALUE
 ACTUAL      :  CENTS     :    $        :  VALUE       :      $        :       $        :
 12345       *   1234500 *    12345.00  *  $ 12345.00 *   12,345.00  *  12,345.00     * $12,345.00
 123456.78   *  12345678 *   123456.78  * $ 123456.78 *  123,456.78  * 123,456.78     * $123,456.78
-1.236       *      -124 *       -1.24  *      $-1.24 *       (1.24) *       1.24 CR * $1.24 CR
-12341.763   *  -1234176 *   -12341.76  *  $-12341.76 *  (12,341.76) *  12,341.76 CR * $12,341.76 CR
 45.469      *      4547 *       45.47  *     $ 45.47 *       45.47  *       45.47     * $45.47
 0           *         0 *        0.00  *     $ 0.00  *        0.00  *        0.00     * $0.00
-10.111      *     -1011 *      -10.11  *     $-10.11 *      (10.11) *       10.11 CR * $10.11 CR
 122.987     *     12299 *      122.99  *     $ 122.99 *      122.99  *       122.99    * $122.99
-555.5211    *    -55552 *     -555.52  *    $-555.52 *      (555.52) *      555.52 CR * $555.52 CR
 .11678      *        12 *         .12  *       $ .12 *         .12  *         .12     * $.12
 909.995     *     91000 *      910.00  *    $ 910.00 *      910.00  *       910.00    * $910.00

UNROUNDED TOTAL =  123971.717
  ROUNDED TOTAL =  $123,971.73
```

Prints Charming

by Andy Gamble

Add some pizazz to your programs. For all Commodore computers.

I'm bored. I'm so very bored with dull programs. You know which ones they are: their presentation is unexciting, remarkably static or just plain dull. Of course you would be annoyed if you paid good money for programs like these, but why settle for less in the programs you write yourself? Fortunately help is at hand, and from an unlikely source.

Watching TV the other evening, a rare thought struck me. It was during one of those movies that make commercials seem interesting and something to look forward to. Suddenly I realized that the commercials were indeed better than the film, and were probably better than MOST films in one sense: their graphic messages. I started flipping through the channels, looking for commercials and making notes and drawings of the graphic techniques they used.

Of course this should come as no big surprise. Advertising is a huge industry and nothing less than perfect sells a product. Computer graphics have had a large effect on the way we expect messages to be presented and the advertising industry has been quick to pick this up and develop the technique. You can now see very impressive graphics everywhere, from commercials for dog food to the intro to the news.

Now I'm not suggesting that your own home computer can necessarily mimic the combined talents of a Michaelangelo and a "Tron" programmer, but amazingly effective graphics can be produced with a minimum of bother. Bear in mind that these are impossible to demonstrate on the printed page, so you are urged to enter the following sample routines into your machine and RUN them. Somewhat apologetically I call them "Prints Charming".

First of all, note that the string to be charmingly printed is A$ in all that follows. The examples also make use of the string functions MID$, LEFT$ and RIGHT$. Perhaps this is a good place to review these functions.

MID$, LEFT$ and RIGHT$ are most often used to isolate single characters or groups of characters from a string, and this is the case for these programs. In greater detail:

LEFT$(A$,X) isolates the X "left-most" characters of the string A$, so that if A$ = SINISTER, LEFT$(A$,3) would return "SIN".

RIGHT$(A$,X) does the same for the "right-most" characters: if A$ = DEXTROUS, RIGHT$(A$,2) returns "US".

MID$(A$,X,Y) is slightly more complicated: It chooses Y characters beginning at character number X. Assigning A$ = KERNEL, MID$(A$,2,3) would return "ERN". In these examples the Y value is mostly one, meaning that we are choosing one character from the middle of the string.

Incorporating any of these little gems into your own programs is child's play. Rather than print out the CBM cursor control characters, I've used the following conventions:

[CU] is Cursor Up
[CD] is Cursor Down
[CL] is Cursor Left
[CR] is Cursor Right
[RVS] is Reverse Video On

You should of course press the appropriate cursor control key instead of the above codes.

The first one I call "Slow Motion Print" (which is actually more exciting than it sounds). Try this small routine:

```
100 A$="SLO-MO PRINT"
110 FOR I=1 TO LEN(A$)
120 FOR ZZ=1 TO 200: NEXT ZZ
130 PRINT MID$(A$,I,1);: NEXT I
140 PRINT
```

Line 120 controls the speed of printing, commonly called a "delay loop". Numbers lower than 200 will speed up the printing and vice versa. The value 200 is about right for normal reading speed.

Here's "Sliding Print", which lives up to its name:

```
100 A$="SLIDING PRINT"

110 A$="           "+A$

120 FOR I=1 TO LEN(A$)

130 PRINT MID$(A$,LEN(A$)-I+1,
    I);"[CU]"
140 FOR ZZ=1 TO 25: NEXT ZZ:
    NEXT I
```

The blanks in line 110 pad out the rather short string so that it ends up in the middle of a 40-column screen. (You might not need these for a longer message or for a VIC screen.)

"Word Crash" is similar: Letters crash into each other, traveling left to right.

```
100 A$="WORD-CRASH"

110 FOR I=LEN(A$) TO 1 STEP-1

120 FOR ZZ=1 TO 25-LEN(A$)+I

130 PRINT TAB(ZZ)" "MID$(A$,I,
    1);"[CU]"
140 NEXT ZZ

150 NEXT I
```

Change the 25 in line 120 to a smaller number (say 10) if you are using a VIC.

I have to admit that I was stuck for a name for this next one: Since it seems as though the string is zooming out of the middle of the screen from hyperspace, in deference to all space cadets I called it "Hyper-Print":

```
100 A$="THIS IS A DEMO OF
    HYPER-PRINT"
110 A=LEN(A$): IF A/2 <> INT
    (A/2) THEN A$=" "+A$:
    GOTO 110
120 FOR I=1 TO A/2

130 PRINT TAB(21-I); LEFT$(A$,
  I); RIGHT$(A$,I)
140 PRINT"[CU]";

150 FOR ZZ=1 TO 50: NEXT ZZ

160 NEXT I
```

Line 110 expands the string to an even number of characters, if need be (the string is divided into two parts for the zooming). We do not use the MID$ function this time but rather the LEFT$ and RIGHT$ functions. Again, line 150 merely slows down the effect a little. "Hyper-Print" works best on longer strings of 20 characters or more. Again, change the 21 in line 130 to about 10 for a VIC, and use short strings for A$.

"Slo-Under" is my name for this next effect, which is very useful for emphasizing parts of a text at reading speed. As you'll see, it underlines words slowly.

```
100 A$="THIS IS A DEMO OF 'SLO-
    UNDER'. LET'S HOPE IT
    WORKS!"
110 L=40: IF LEN(A$)<40 THEN
    L=LEN(A$)
120 PRINT LEFT$(A$,L)

130 IF L=40 THEN PRINT"[CU]";
```

```
140 FOR I=1 TO L

150 PRINT CHR$(163);

160 FOR ZZ=1 TO 30: NEXT ZZ

170 NEXT I

180 A$=RIGHT$(A$,LEN(A$-L)

190 IF A$<>"" THEN 110
```

In this program A$ is split into lengths of 40 characters (the width of the PET screen) by lines 110,120 and 180. These sub-strings are then printed with a blank line between each, which is where the underline character goes (163 in line 150). The process continues as long as there is a string A$ left (line 190). Change the 40 in lines 110 and 130 for the VIC screen to 22.

"Alpha-Print" is likewise harder to explain than to type. For each letter of the message the alphabet whizzes by, stopping at the correct character:

```
100 A$="ALPHA-PRINT"

110 FOR I=1 TO LEN(A$)

120 M=ASC(MID$(A$,I,1))

130 IF M<65 OR M>90 THEN PRINT
    CHR$(M);: NEXTI
140 FOR J=65 TO M

150 PRINT CHR$(J);"[CL]";

160 NEXT J

170 PRINT"[CR]";

180 NEXT I
```

In CBM ASCII the letter A is 65 and Z is 90: Line 130 traps all non-letters and prints them directly.

"Twinkle-Print" alternates characters in the string between normal and reverse video at random:

```
100 A$="TWINKLE-PRINT"

110 PRINT A$

120 ZL=LEN(A$): DIM ZS(ZL)

130 ZR=INT(ZL*RND(1)+1)

140 PRINT"[CU]"; TAB(ZR-1);

150 IF ZS(ZR)=0 THEN ZS(ZR)=1
    :PRINT"[RVS]"; MID$(A$,
    ZR,1):
GOTO 130

160 IF ZS(ZR)=1 THEN ZS(ZR)=0
    :PRINT MID$(A$,ZR,1):
    GOTO 130
```

An array ZS( ) is set up to show the normal/reverse state of each character (0 or 1). Characters chosen at random by ZR in line 130 are then "flipped".

The last example makes the individual letters move up and down one line (they look as if they're bouncing). The technique is almost the same as the previous one, but definitely much sillier.

```
100 A$="BOUNCE-PRINT"

110 PRINT A$

120 ZL=LEN(A$): DIM ZS(ZL)

130 ZR=INT(ZL*RND(1)+1)
```

programmer's tips

```
140 PRINT"[CU]"; TAB(ZR-1);

150 IF ZS(ZR)=0 THEN ZS(ZR)=1:
    PRINT"[CU]";
MID$(A$,ZR,1);"[CD][CL]";" ":
    GOTO 120
160 IF ZS(ZR)=1 THEN ZS(ZR)=0:
    PRINT
MID$(A$,ZR,1);"[CU][CL]";" ";"
    [CD]": GOTO 120
```

Note that there are single spaces between quotes in lines 150 and 160.

And finally a word of caution: Printing charmingly does not necessarily make a good program. As in all good things, a little goes a long way. But using one or two of these effects in your programs could brighten them up no end. And don't forget that very useful source of ideas, the TV commercial. Soon you'll be getting up to make some coffee during the movie, and not during the breaks... C

COMAL Graphics

by Len Lindsay

The COMAL programming language is available in the public domain for Commodore's PET/CBM and Commodore 64 computers. Here the author of the COMAL Handbook *explains COMAL's LOGO-like graphics capabilities.*

The Commodore 64 is a very fine computer. However, controlling its many advanced graphics features can be cumbersome if you use only the BASIC that comes with the computer. COMAL is one advanced programming language that takes up where BASIC left off. It adds, among other things, a turtle graphics system as well as sprite controls. In my last article, I explained COMAL as a programming language. This article will briefly explain COMAL as a graphics language.

Turtle graphics were popularized by the LOGO programming language. In fact, turtle graphics are often incorrectly called LOGO graphics. With the introduction of COMAL for the Commodore 64, users now have two fine languages, both of which include a similar turtle graphics system. Both LOGO and COMAL are disk loaded. However, LOGO is on a protected, copyrighted disk, while COMAL is public domain.

I have compiled several charts to help illustrate and explain COMAL graphics. First, I would like you to look at the chart comparing the turtle graphics systems of LOGO and COMAL. Notice that they are very similar; both are based on the turtle graphics in M.I.T. LOGO. The chart also illustrates the few differences between the two systems. Next, you can refer to the chart of COMAL graphics keywords. This is a fairly complete list of all the keywords relating to the graphics system, including syntax and examples.

Turtle Graphics

Most of us are familiar with graphics using the X,Y coordinate plotting system. COMAL graphics includes this familiar system. This system, however, is a very rigid, absolute system. After alot of research, M.I.T. (Massachusetts Institute of Technology) came up with the more flexible, relative graphics system now called turtle graphics.

The idea of using a "turtle" in a graphics system may seem a bit strange at first, but once you have tried it, you will understand its significance. It is not restricted only to children's pictures. It can be used for serious charting as well.

Imagine a turtle sitting in the middle of your screen. It has 16 different colored pens. It is "holding" one of the pens. If this pen is "down", whenever the turtle moves it will leave a line behind of that pen's color. However, if the pen is "up", no line will be drawn, since the pen is not down on the screen. You have full control over the turtle's movements. In COMAL, you can even vary the size of your turtle from small to big, with ten different sizes. And if you don't want to see the turtle, you can "hide" it, making it an invisible turtle. It still will move around and draw lines, but you will not be able to see it.

Moving the Turtle

Moving the turtle is a relatively simple matter. Movement is in "units". If you want the turtle to move forward 50 units, simply enter:

```
FORWARD 50
```

If you want the turtle to move back 15 units, you would enter:

```
BACK 15
```

You can also "put" the turtle at any specific X,Y location you want. To put it at X,Y coordinates 160,100 you would enter:

```
SETXY 160,100
```

Turning the Turtle

Your turtle can move forward and back, but not sideways (except using the SETXY command). To move the turtle sideways, you first "turn" the turtle either left or right. The turning is specified in "degrees", with 90 being a right angle and 360 being a complete turn, back to the same heading as the turtle started with. To turn the turtle left 90 degrees simply enter:

```
LEFT 90
```

To turn the turtle right 1 degree, you would enter:

```
RIGHT 1
```

While most of the time you will turn your turtle with either a LEFT or RIGHT command, you also can tell it to face any specific absolute direction. To set its heading to 180 degrees, you would enter:

```
SETHEADING 180
```

The Turtle's Pen

The turtle is always holding one pen. It can be one of 16 colors. These colors are numbered in the same manner as listed in the Commodore 64 users manual (see the chart on page93). If you would like the turtle to be holding a white pen (white is color 1), simply enter:

```
PENCOLOR 1
```

In order for the turtle to leave lines behind while it moves, you must tell it to put its pen down:

```
PENDOWN
```

If you don't want your turtle to draw lines, simply enter:

```
PENUP
```

Background

You can set the background and border colors easily with COMAL graphics. To set the color of the background to black (color 0) simply enter:

```
BACKGROUND 0
```

To set the border to red (color 2) you would enter:

```
BORDER 2
```

Try It

You now know enough about your turtle to draw a picture on the screen. The turtle's pen and the background and border all have default settings. We will use these settings in our first examples simply because they are good settings and the examples will be easier.

To go into "graphics mode" you simply enter the command DRAW. Your turtle starts in the center of the screen with its pen down. It is pointing straight up. The background is black, the border is black, and the pen is white.

Now, let's have the turtle draw a simple box that is 40 units square. Issue the following commands:

```
FORWARD 40
LEFT 90
FORWARD 40
LEFT 90
FORWARD 40
LEFT 90
FORWARD 40
LEFT 90
```

I hope you noticed that you issued a FORWARD and LEFT command sequence four times. You did this in direct mode while in the graphics mode (i.e., the DRAW command issued previously). We could write a short COMAL program to do the same thing, but we would write the program in the text mode. To go into text mode simply issue the command TEXT.

```
10 PROC SQUARE
20   FOR SIDE=1 TO 4 DO
30     FORWARD 40
40     LEFT 90
50   ENDFOR SIDE
60 ENDPROC SQUARE
```

Notice that COMAL lets you define your own procedures. In user-defined procedures, the parameters are enclosed in parentheses. Thus you enter SQUARE(90) instead of SQUARE 90. The COMAL system will remember any procedures and functions in your program after it is run as well as while it is running. Thus once you have entered your program (such as the procedure SQUARE above) you must issue the command RUN to "initialize" your user-defined procedure. Then you can use the SQUARE command from direct mode as often as you want, as long as you don't change the program.

Remember that while in text mode, the turtle will still follow your commands, but you will not be able to see it, or anything it does. To go back to the graphics screen you issue the command DRAW. Now, once your program is run you can draw a square anytime by giving the command SQUARE.

Drawing a square that is the same size all the time can be quite boring. We can use a variable with any of the turtle commands. COMAL also allows us to pass values into a procedure (called PROC). Thus we could write a flexible SQUARE procedure like this:

```
10 PROC SQUARE(LENGTH)
20   FOR SIDES=1 TO 4 DO
30     FORWARD LENGTH
40     LEFT 90
50   ENDFOR SIDES
60 ENDPROC SQUARE
```

Now, to have the turtle draw a square, simply issue the command (remember to RUN the program first and to be in graphics mode):

```
SQUARE(80)
```

A Familiar Design

Now let's set up a flexible yet simple design procedure. The SQUARE procedure had the length of a side as its variable and always used 90 as the degrees to turn. Now let's have the number of degrees to turn to be the variable and have the length of the side increase by one after each side is drawn:

```
PROC DESIGN(ANGLE)
  CLEARSCREEN
  FOR SIDE=1 TO 99 DO
    FORWARD SIDE
    RIGHT ANGLE
  ENDFOR SIDE
ENDPROC DESIGN
```

To see some designs give these commands:

```
DESIGN(89)
DESIGN(91)
DESIGN(59)
```

A Complete Squares Program

Now we can write a complete program to draw some squares on the screen. To use this program we only need to issue the command RUN. Everything is done automatically after that. First make sure you are in text mode:

```
NODRAW
```

Next make sure to clear out any previous program:

```
NEW
```

Now, to make program writing easier, use automatic numbering:

```
AUTO
0010 BACKGROUND 0
0020 BORDER 0
0030 DRAW
0040 REPEAT
0050   PENUP
0060   SETXY RND(0,320),
       RND(0,200)
0070   PENDOWN
0080   SETHEADING RND(0,359)
0090   PENCOLOR RND(1,15)
0100   SQUARE(RND(5,50))
0110 UNTIL KEY$<>CHR$(0)
0120 //
0130 PROC SQUARE(LENGTH)
0140   FOR SIDES=1 TO 4 DO
0150     FORWARD LENGTH
0160     LEFT 90
0170   ENDFOR SIDES
0180 ENDPROC SQUARE
0190 just hit return here
```

To look at your program simply tell the computer to list it for you:

```
LIST
```

Now, save your program in your disk:

```
SAVE "SQUARES"
```

Finally, run the program:

```
RUN
```

The program will now draw squares that are a random color, random size, at random angles in random locations on the screen. It will continue until you press any key.

Start Playing

You are now ready to start playing with your new turtle friend. You will be surprised how many things you can draw. As you define design commands, put

them into a procedure. Then you can draw that design at any time simply by calling it by name.

Next Time

Next time I will cover manipulating sprites using COMAL. The procedure is very easy since COMAL has built-in sprite commands—no more PEEKs and POKEs to have some spritely fun. **C**

COMAL
Graphics Keywords

BACK <length>
BACK 40 moves turtle backwards 40 units
moves turtle back the number of units specified

BACKGROUND <color number>
BACKGROUND 2 sets the background to red (2)
sets background color to the one specified

BORDER <color number>
BORDER 2 sets the border to red (2)
sets the border color to the one specified

CLEARSCREEN
CLEARSCREEN clears graphic screen—turtle is not affected

CLEARTEXT
CLEARTEXT clears text screen

CORRECTION <percentage>
CORRECTION .87
corrects for your screen so a circle is a circle not an oval

DRAW
DRAW go into graphics screen

DRAWTO <x coordinate>,<y coordinate>
DRAWTO 45,80 draws a line from current point to 45,80
draws a line to point specified (x/y graphics—not turtle)

FILL <x coordinate>,<y coordinate>
FILL 25,80 fills in the area that includes point 25,80
fills the area containing the specified point with current color

FORWARD <length>
FORWARD 40 moves turtle forward 40 units
moves turtle forward the number of units specified

FRAME (<x0>,<x1>,<y0>,<y1>)
FRAME(0,320,0,200) sets the sides of the screen: horizontally from 0 to 320 and vertically from 0 to 200
sets the four sides of the screen

FULLSCREEN
FULLSCREEN goes to a fullscreen graphics screen

GETCOLOR (<x coordinate>,<y coordinate>)
GETCOLOR(25,80) returns the color number of the point 25,80
returns the color number of the specified coordinate point

HEADING
HEADING returns the current turtle heading number in degrees

HIDESCREEN
HIDESCREEN turns screen off without disturbing its contents

HIDETURTLE
HIDETURTLE make the turtle invisible

HOME
HOME put the turtle in its home position

LEFT <degrees>
LEFT 90 turn left 90 degrees
changes turtles heading by specified degrees to the left

MOVETO <x coordinate>,<y coordinate>
MOVETO 50,80 moves to point 50,80 without drawing a line
moves to specified point without leaving a line (x/y graphics)

NODRAW
NODRAW goes to the text screen

PENCOLOR <color number>
PENCOLOR 2 sets the current pen color to red (2)
changes the color of the turtle and its pen

PENDOWN
PENDOWN puts the pen down so a line is drawn when turtle moves

PENUP
PENUP picks up pen so no line is drawn when turtle moves

PLOT <x coordinate>,<y coordinate>
PLOT 35,70 plots a point at location 35,70 in current color
plots a point in the current pencolor at specified coordinates

PLOTTEXT <x coordinate>,<y coordinate>, <text$>
PLOTTEXT 16,24,"TESTING" prints TESTING on graphics screen
prints text onto the graphics screen starting at x,y specified

RIGHT <degrees>
RIGHT 90 turns turtle 90 degrees to right (a right angle)
changes turtle's heading by specified degrees to the right

SCALE <x0>,<x1>,<y0>,<y1>
SCALE −500,500,−200,200 sets the screen to 'cover' points horizontally from −500 thru 500 and vertically from −200 through 200
sets the coordinates of the four sides of the screen

SETHEADING <degree>
SETHEADING 0 sets the turtle's heading to 0 degrees
changes turtle's heading to that heading specified

SETX <x coordinate>
SETX 20 positions turtle with same y coordinate and 20 x coordinate
sets the turtle's x coordinate to the one specified

SETXY <x coordinate>,<y coordinate>
SETXY 20,50 positions the turtle to location 20,50
sets the turtle's x and y coordinates as specified

SETY <y coordinate>
SETY 30 positions turtle with same x coordinate and 30 y coordinate
sets the turtle's y coordinate to the one specified

SHOWSCREEN
SHOWSCREEN turns the screen on

SHOWTURTLE
SHOWTURTLE the turtle is now visible

SPLITSCREEN
SPLITSCREEN goes to graphic screen with 2 line text area

TURTLESIZE <size>
TURTLESIZE 6 sets the size of the turtle to 6
vary the size of turtle from small (0) to large (10)

XCOR
XCOR returns the x coordinate of the turtle

YCOR
YCOR returns the y coordinate of the turtle **C**

Commodore 64 COMAL Colors List

Color Value	Color Name	Color Value	Color Name
0	Black	8	Orange
1	White	9	Brown
2	Red	10	Light Red
3	Cyan	11	Dark Grey
4	Purple	12	Medium Grey
5	Green	13	Light Green
6	Blue	14	Light Blue
7	Yellow	15	Light Grey **C**

Commodore 64 COMAL Resource List

Available from COMAL Users Group, 5501 Groveland Terrace, Madison, WI 53716 (608) 222-4432:

COMAL on disk with programs, $15
COMAL Handbook, by Len Lindsay, $19
Turtle Sourcebook, by Jim Muller, $22
Newsletter, Info, List of local user groups, $2

Available from Educational Company of Ireland Ltd., Ballymount Road, Dublin, 12 Ireland:

Foundations in Computer Studies With COMAL, by John Kelly

Available from Ellis Horwood Limited, Market Cross House, Cooper Street, Chichester, West Sussex PO19 1EB England:

Beginning COMAL, by Borge Christensen
Structured Programming With COMAL, by Roy Atherton **C**

Turtle Graphics: LOGO and COMAL

Items	CBM LOGO	CBM COMAL
Turtle Control:		
Move forward length	FORWARD	FORWARD
Move to a point	SETXY	SETXY
Move only x coordinate	SETX	SETX
Move only y coordinate	SETY	SETY
Move backward length	BACK	BACK
Home turtle	HOME	HOME
Turn turtle left	LEFT	LEFT
Turn turtle right	RIGHT	RIGHT
Turn to specific heading	SETHEADING	SETHEADING
Make turtle visible	SHOWTURTLE	SHOWTURTLE
Make turtle invisible	HIDETURTLE	HIDETURTLE
Pen up off paper	PENUP	PENUP
Pen down on paper	PENDOWN	PENDOWN
Set pen color	PENCOLOR	PENCOLOR
Pen to erase	PENERASE	PENERASE
Number of colors	16	16
Colors set by NAME/NUM	by number	by number
Set size of turtle	—	TURTLESIZE
Plot a point	—	PLOT
Values Returned:		
Get color of point	—	GETCOLOR
Get X position of turtle	XCOR	XCOR
Get Y position of turtle	YCOR	YCOR
Get current heading	HEADING	HEADING
Screen and Color Control:		
Clearscreen	CLEARSCREEN	CLEARSCREEN
Clear text screen	CLEARTEXT	CLEARTEXT
Set to graphics mode	DRAW	DRAW
Set to text screen	NODRAW	NODRAW
Set a screen boundary	—	FRAME
Set background color	BACKGROUND	BACKGROUND
Set border color	—	BORDER
Fill in an area	—	FILL
Full screen mode	FULLSCREEN	FULLSCREEN
Split screen mode	SPLITSCREEN	SPLITSCREEN

C

Promqueen/64

Reviewed by Jeff Bruette

The Promqueen/64 is both a device that allows you to read and write EPROMs and a software development tool. It is for use on the Commodore 64 and the SX-64 computers. The PQ/64 is manufactured by the Gloucester Computer Company, the same company that developed the original Promqueen for the VIC 20.

With the Promqueen/64 you can write programs then put them on EPROMs. An EPROM, or Erasable/Programmable Read Only Memory, is a chip that retains a program in memory even after the power has been turned off. The only way to erase the program is by directing ultraviolet light onto the EPROM. With the PQ/64 you can create programs for the Commodore 64, VIC 20 or any other device that uses EPROMs. In addition to the hardware, the PQ/64 comes with a utility called HEXKIT.

The PQ/64 plugs directly into the cartridge port. On top, it has a ZIF (zero insertion force), 28-pin socket. There is a mode switch that allows the PQ/64 to be in any of three states. The modes are "off", "copy" and "burn". The most intriguing switch is what is called a "matrix switch". This switch is used to select the type of EPROM that the PQ/64 will be dealing with. Because of this switch there is no need for the "personality modules" used in most other EPROM burners.

The PQ/64 can read and write to many different types of EPROMs. This list includes 2516, 2532, 2564, 2716, 27C16, 2732, 27C32, 2764, 2758 and 27128. In addition, the PQ/64 was designed with future developments in mind. This means that when 32K byte EPROMS become available the PQ/64 will be able to handle them. There is a potentiometer which can be adjusted to select the necessary voltage for the EPROM that is being used. One other feature is a reset button that can be used to "cold start" the computer without erasing the RAM.

The HEXKIT utility software that is supplied with the PQ/64 is a useful development tool. It comes on an EPROM which, using HEXKIT itself, can be saved on disk or tape. In HEXKIT there is an edit mode, which lets you look at and modify any memory location in RAM. In addition, when programming, you can specify labels that can be used in jump vectors. When debugging a program, you can set breakpoints so your program will stop at these points. Once your code has been written you have the option to save it. You can save to disk or tape as well as sending your program to the RS232 port. If you would like a printed copy, your code can be saved to the printer.

Your next step would be to make an EPROM of your program. After setting the matrix switch and voltage to match the EPROM that is being used, put the EPROM in the ZIF socket. When the option to burn is selected, you are asked where the code that is to be burned is located. This is useful because different programs may be in different memory locations. After you enter the address range a check is made to verify that a blank EPROM is being used. If the EPROM is not blank or is bad, the software brings this to your attention. This is quite handy since a blank EPROM looks just like an EPROM that has a program on it.

If everything is OK, the programming of the EPROM begins and the screen displays what address is being burned. After the PQ/64 has burned the program into the EPROM it verifies that the code that is in RAM matches the code on the EPROM. If, for some reason, there is an error, the screen displays the address at which the error occurred.

In conclusion, it is my opinion that the Promqueen/64 is a necessary tool for developers as well as hobbyists who wish to learn more about programming and hardware. On a scale of one to ten, I'd rate Promqueen/64 like this:

Quality:	9
Ease of use:	9
Documentation:	8
Usefulness:	10
Price/Performance:	9
Support:	10
Average:	9.1

C

A Theory of Operation for the VIC/64 Boards

by Jim Gracely

The "board" inside your computer is laid out in "functional blocks". Here is a simple explanation of what each block does.

How does the VIC or 64 work? This is an interesting question, and it actually has a number of completely different answers. Ask a BASIC programmer and he'll explain to you how the page zero pointers are used and how the BASIC ROM uses the kernal routines. Ask a machine language programmer and he'll explain how interrupts are used and how the microprocessor uses the stack. In this article however, we're going to look at the two computers from a hardware point of view. What exactly is on the computer board inside that plastic case and how does it work?

To explain how the VIC and 64 computer board operates, I am going to use a functional block description. This is accompanied by a functional block diagram and explains how the board can be divided into functions (i.e., memory, I/O etc.). These descriptions are only to increase your understanding of the computer, and are not intended to be a technical training course.

I want to note at this time that these descriptions are going to be for both the VIC and the 64. On the functional level, the two computers are just about identical and in areas where there are differences I will discuss both machines.

Functional Blocks

The computer board on both the VIC and the 64 can be divided into four functional blocks.

1) Control and Timing
2) Data Input/Output
3) Audio/Video Output
4) Memory

Figures 1 and 2 are block drawings of the boards and can be referred to while reading the following descriptions.

Control and Timing

This block controls every other block in the system. It contains the central processing unit, the clock crystal and timing chips.

Central Processing Unit: Here is the heart of both computers. It is this single component which controls all internal functions of the computers. Any program, whether written in BASIC, machine language or Pascal is translated and run by the CPU. The CPU also controls all input and output of data both within the system and to all peripherals.

On the VIC board, the CPU is a 6502 chip. This is the "host chip" for this issue and there are a number of other articles in this issue on it.

On the 64 board, the CPU is a 6510 chip. This is the same as the 6502 chip with one small difference. There are two special input/output registers in the chip. One register is an input/output port. The other register is a data direction register (DDR) and determines whether the bits of the first register are to be used as input or output. One of the special uses of these registers is to communicate with the datassette.

Data Input/Output

This block is the interface between the Control and Timing block and all data-oriented devices. These devices include the keyboard, joysticks, disk drive, cassette and printer.

On the VIC board, this block contains two VIA (Versatile Interface Adapter) chips. VIA #1 communicates with the keyboard and the cassette. It also provides output data to the serial port (disk drive or printer). VIA #2 communicates with the user port and joystick and accepts the input data from the serial port.

On the 64 board, this block contains two CIA (Complex Interface Adapter) chips. CIA #1 communicates with the keyboard and both joystick ports. CIA #2 communicates with the user port and the serial port (disk drive or printer).

Audio/Video Output

This block is the interface between the Control and Timing block and a TV or monitor. This block also communicates with the Memory block and the Input/Output block.

In the VIC this block contains the VIC (Video Interface Controller) chip. This chip accepts paddle input data and provides the audio and video signals for the video/audio port.

In the 64 this block contains the

VIC II chip, the SID (Sound Interface Device) chip and an RF (Radio Frequency) modulator. The VIC II chip provides video output to both the RF modulator and the audio/video port. The SID chip accepts input from paddles and provides audio output for the RF modulator and the audio/video port. The SID chip also accepts an audio input from the audio/video port. The RF modulator "modulates" the audio and video signals so that they can be used by a television.

Memory

This block contains all of the RAM (Random Access Memory) and ROM (Read Only Memory) that is available in the computer.

Both the VIC and the 64 contain two 8K X 8 ROM chips (one for BASIC and one for the kernal). Both also contain a 4K X 8 character ROM chip and have a 1K X 4 color RAM chip. The VIC contains ten 1K X 4 RAM chips for a total of 5K bytes of RAM, part of which is for the screen and BASIC working storage (0-1023). The 64 contains eight 64K X 1 RAM chips for a total of 64K bytes of RAM. **C**

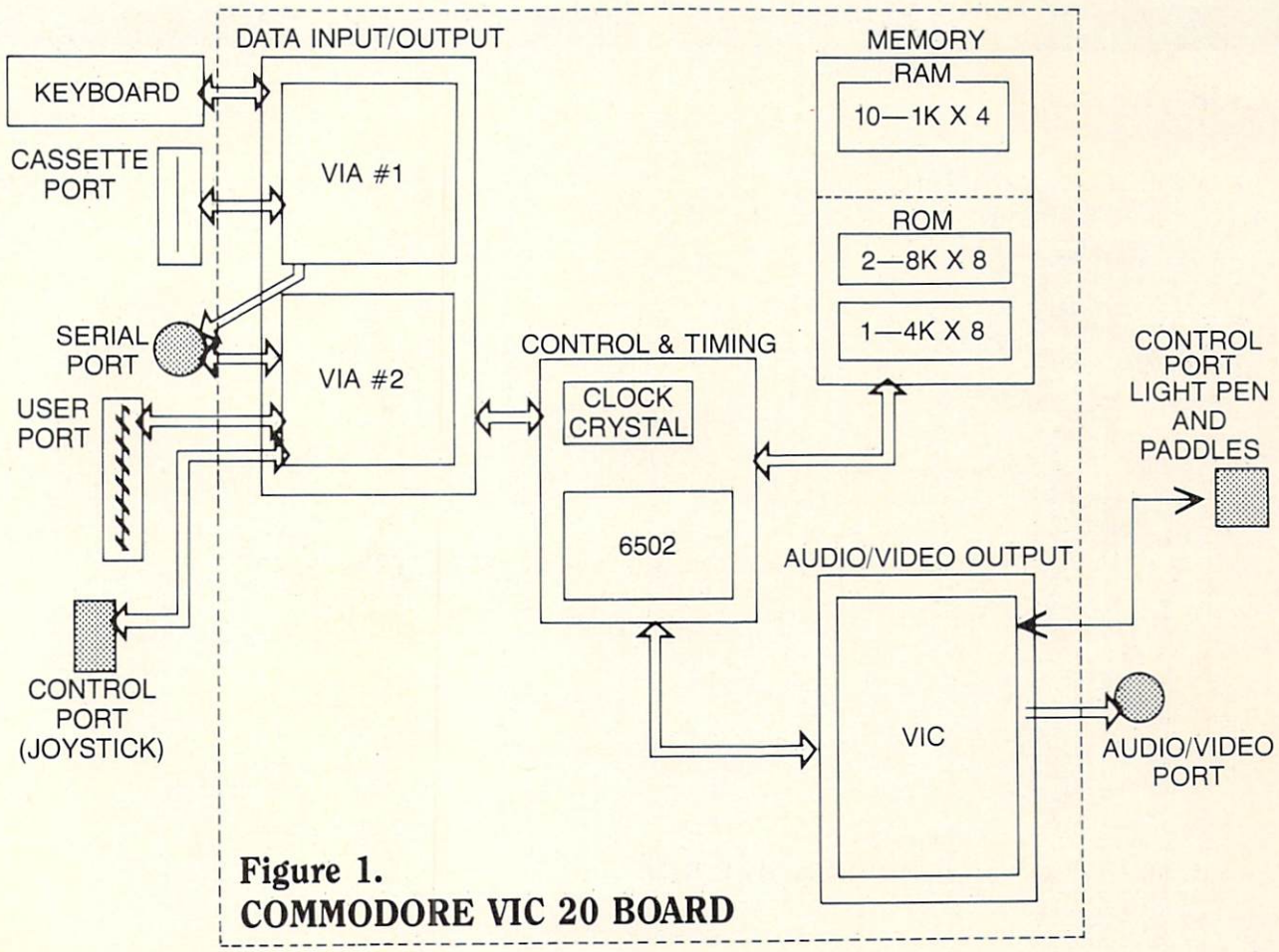


Figure 1.
COMMODORE VIC 20 BOARD

technical

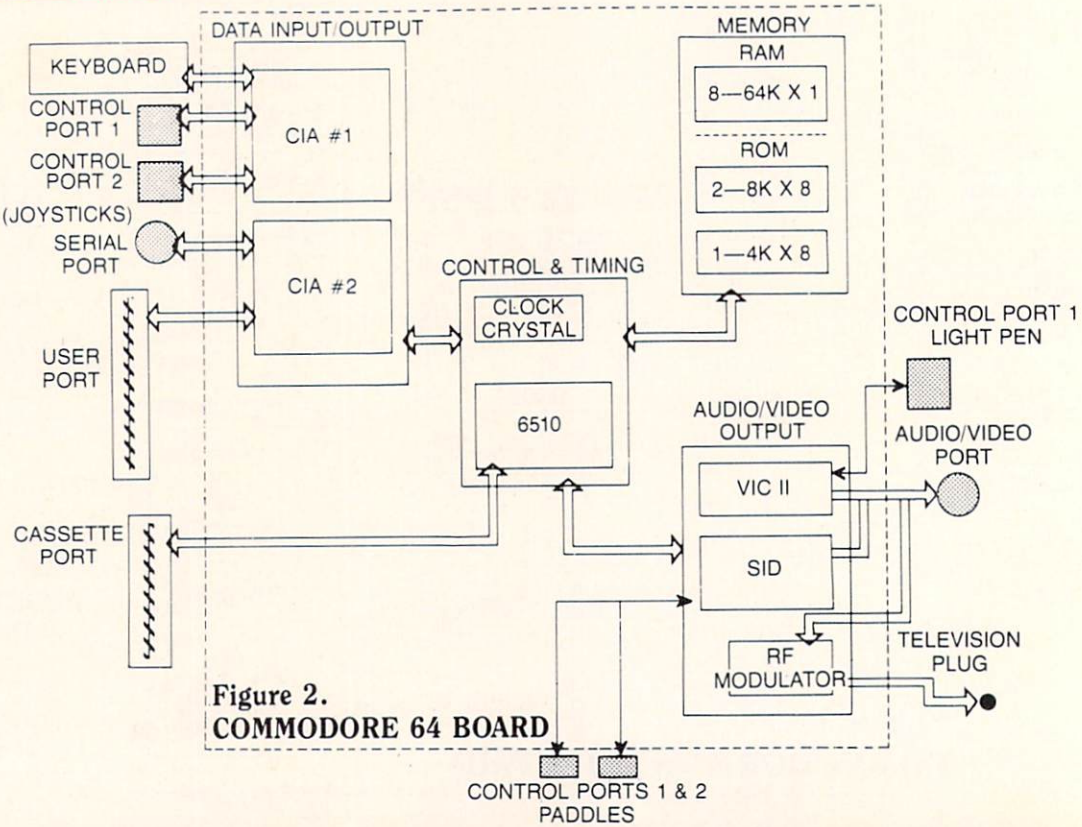


Figure 2.
COMMODORE 64 BOARD

The New Commodore Information Network

by Barbara Karpinski

Commodore's telecommunications network has expanded.

Many of you may have heard of the Commodore Information Network (CIN) on CompuServe. You may have read about CIN either as part of a snap-pak offer with the VICMODEM or AUTOMODEM or in the March, 1983, *Commodore* magazine article by Jeff Hand. BUT... if you haven't accessed CIN lately, you haven't seen anything yet. Since that last March update, CIN has grown and expanded rapidly. On page 100 is an outline of the new CIN.

Bulletin Boards

One of the most noticeable areas of rapid growth is in our National Bulletin Board (BBS). We have split our Commodore BBS Interest Group into three separate BBS's—one for the VIC 20 computer, one for the Commodore 64, and one dedicated to our CBM business machines (CBM/PET, SuperPET and "B" Machine).

One good analogy for a BBS is to think of it as a club. If you want, you can become a member and participate in all the club has to offer or you can just look around to see if you want to join. Like any special interest club, our three BBS's offer different topics of interest depending on what members want to see. These topics are contained in either the SIG (Special Interest Group) area, the conference area (CO) or in the databases of each BBS.

In the SIG, you can communicate with other members by posting either private or public messages. In our CO area you can set up online conferences or attend one of the many that CBM sponsors. The third feature of every BBS is the databases. In these areas are stored files, programs, errata, answers to technical questions and lots more.

VIC 20 Bulletin Board

Some topics of interest in this SIG are:

1. HOTLINE answers to frequently asked questions about the VIC 20.
2. A large collection of application and recreation programs.
3. A library of past *Commodore* and *Power/Play* magazine programs.
4. Weekly conferences on popular topics.

Commodore 64 Bulletin Board

These are some of the new topics in this SIG:

1. The whole collection of educational public domain programs.
2. A "New Languages/FORTH" section where members who are interested in this topic can share ideas and program knowledge.
3. An entire section dedicated to CBM CP/M® programs. (CP/M is a registered trademark of Digital Research, Inc.)
4. A lot of music programs including the Entertainer.
5. Conferences are held in this SIG weekly.

CBM Business Machines Bulletin Board

Business-oriented machines are featured here, notably the CBM/PET, SuperPET and the "B" machine.

1. One section is reserved for each of the three computer lines mentioned above.
2. An Educational area is set up for all to use.
3. Information and errata on the business machines can be found here.

Explanations of the commands to get around in each area of the Bulletin Boards can be found in the March, 1983, issue of *Commodore* magazine or online by either typing "H" at any [! or FUNCTION:] prompt or by going to our Survival Kit choice [go CBM4] at the [! or FUNCTION:] prompt.

Main Menu Update

Our main menu or display area, which has been outlined above, has also been updated. Perhaps the most exciting addition to this menu is the NEW UPDATES TO CIN choice. When you choose this option you will see which area has been updated and the date. In this way you do not have to keep scanning each videotex page for recent changes. All you need do is to periodically check this area to see

Commodore Information Network's Main Menu

- Intro & Survival Kit
  - -Direct Access Page Numbers
  - -Videotex Area Explanation
  - -Bulletin Boards Expl.
  - -HOTLINE Instructions
- New Updates to CIN
- HOTLINE Menu
  - -Introduction
  - Self-Help Files
  - -HOTLINE (Ask Questions)
- Product Announcements
- Bulletin Boards Menu
  - -VIC 20 Users Group
  - C-64 Users Group
  - -CBM Bus Machines Users Group

  Each BB Composed of:
  1. SIG
  2. Conference
  3. Databases
- Magazine Articles
  - -*Commodore: MicroComputer* Mag.
  - -*Power/Play* Mag.

  Preview Abstracts of upcoming articles.
- Directories
  - -Dealers across USA & Canada
  - -User Groups across USA & Canada
- Commodore Tips
  - -Software Tips
  - -Hardware Tips
  - -Computers
- CBM Product Line-
  - Hardware
    - -VIC 20/C-64
    - -PET/CBM
  - Software
    - -For VIC 20
    - For C-64
    - -For PET/CBM
  - Mag./Books/Manuals
  - -Accessories
- User Questionnaire
  - -Introduction
  - SIG Explanation
  - Database Explanation
  - Conferencing
  - Vendor Advertising
  - -Uploading/Downloading
  - -VIC 20 Material
  - C-64 Subject Matter
  - Disk Drive Info.
  - Printer Information
  - 8000/9000 Series
  - Software Material
  - Modem Material
  - Monitor
  - -Miscellaneous

if we have made any changes to CIN since the last time you were on. This option will also save you from wasting precious time that can be spent for more important matters such as downloading new programs and responding to messages.

Videotex or Display Area

Below are the ten main choices in the display (videotex) area and a short explanation of each.

CBM-1—This is the page number of our main menu. To get to any area in CompuServe you can search by going through the menu structure choices or by typing in the appropriate page number of the topic. Following is a direct access index guide for a few of the topics that can be found in CIN.

Topic	Page Number
Main Menu	cbm-001
Intro/Survival KIT	cbm-004
Direct Access Index	cbm-021
Videotex Explanation	cbm-024
Bulletin Boards Menu	cbm-029
HOTLINE Explanation	cbm-041
Database Explanation	cbm-047
HOTLINE Menu	cbm-964
HOTLINE (ask questions)	cbm-200
Product Announcements	cbm-007
Bulletin Boards Menu	cbm-006
VIC 20 BB	cbm-962
C-64 BB	cbm-963
CBM Bus. Machine BB	cbm-310
Commodore Magazines Menu	cbm-009
Commodore magazine	cbm-096
Power/Play magazine	cbm-181
Directories Menu	cbm-010
Dealer List	cbm-246
User Groups List	cbm-248
Commodore Tips Menu	cbm-011
Software Tips	cbm-466
Technical Tips	cbm-467
Product Line	cbm-012

1) *Intro/Survival Kit:* A comprehensive user guide for this Network. Includes an explanation of commands.

2) *New Updates To CIN:* Already explained above.

3) *HOTLINE:* A service provided by Commodore to its customers to answer technical and general questions. You can leave a message on the HOTLINE and expect an answer in your electronic mail within several days.

4) *Product Announcements:* (What's New) A listing of new CBM products and events.

5) *Bulletin Boards:* This area was also explained above.

6) *Commodore Articles:* Here you will find abstracts of upcoming articles from CBM magazines.

7) *Directories:* Here you can find an up-to-date list of user groups and dealers across the U.S. and Canada.

8) *Commodore Tips:* Have trouble with programming? Consult our CBM software tips section for new ideas. In addition, our technical tips section has information on how to do neat things with your computer.

9) *CBM Product Line:* A product line listing of all our software and hardware products.

10) *User Questionnaire:* A user questionnaire designed to give CBM a better idea of who uses CompuServe so we can serve you better.

Since nothing in this world stays static, least of the Commodore Information Network, we will periodically be informing you of new updates, additions and changes to CIN—A sort of "what's new" in CIN. So watch for our updates and if you happen to think of a brilliant idea or contribution for our Network, let us know about it. C

The New Model 1650 AUTOMODEM

by James E. Darrough

Commodore's new AUTOMODEM, for the VIC 20 and Commodore 64, provides automatic dialing, automatic answering and more.

Recently Commodore sent me a new product for review, which I am sure will be of interest to those of us into telecommunications. This new product is the Model 1650 AUTOMODEM, now available at your dealer. The AUTOMODEM will operate nicely with the VIC 20 or Commodore 64, although you must presently write your own software to take advantage of autodialing and autoanswering on the VIC. However, I did manage to try out a prototype of VICTERM-40 at NCC, and it will work nicely with the AUTOMODEM. (The VICTERM-40 cartridge should be out within a month or so after this article is published.)

In contrast to the VICMODEM, the AUTOMODEM is a bit larger. It measures about five inches in width and four in depth, although the thickness of the cartridge is about the same. There are, however, vast improvements inside. The three main advantages over the VICMODEM are:

1. Autodial/Autoanswer capability.
2. Switch-select for telephone or modem.
3. Switch-selectable duplex.

Along with the drastic software price decreases by Commodore we have all seen lately, this new autodial/autoanswer modem should put the "icing on the cake." For the suggested retail price of $149.95 I feel this modem falls into the "great deal" category. For one thing, it plugs into either a VIC 20 or the Commodore 64, and comes with all the instructions you will need to write your own terminal programs to utilize the automatic aspects.

A particularly attractive feature of the AUTOMODEM is of course its ability to automatically dial the phone number you select (with software). The modem uses pulse-dialing vice tone-dialing, however, so if you want to access almost any of the long-distance services, you will still have to use your telephone buttons to do so. This is the only real drawback to the AUTOMODEM I have been able to find, and for the price, I feel I can put up with a little button-pushing anyway! For my uses, the AUTOMODEM has worked out very well.

Since I do most of my telecommunicating with CompuServe, I use the autodial function quite often. As far as I know, all the node-numbers for accessing CompuServe are pulse or tone-dial capable, so you can use the modem with no problems to dial-up the service. The AUTOMODEM works equally well when accessing CompuServe or when calling a local database run by a microcomputer, since it follows appropriate standards for data-transfer over telephone lines. In fact, I can't remember ever getting garbage over the phone lines while using the new modem! This speaks well for a low-priced device, since one would expect minor problems with something that costs so little. Commodore has apparently made an effort to preclude this sort of problem with the new modem. Hopefully, the finished user manual for the AUTOMODEM will show equal care (I received a preliminary copy that seemed well-written, so I expect the manual will be readable.)

The AUTOMODEM plugs into the same port as the VICMODEM. As mentioned earlier, it's a little larger in width than a VICMODEM, but still presents a small package, thus avoiding clutter on your table. Even with the standard telephone plugged into the AUTOMODEM, there doesn't seem to be an inordinate amount of wire strung all over the table, which leaves more room for scribbling down those little notes we all use when we don't want to save a whole file.

There are three slide-switches on the AUTOMODEM. Two are mounted on the left side of its case (as you are facing the front of the computer with the modem plugged in), and one is located on the right side. The two on the left are for selecting Answer/Originate (A/O) and for selecting whether you want to use the telephone itself, or the modem (Data/Telephone). This is a nice feature if you want to use the telephone

without going into the other room! It will also prevent you from forgetting that the 64 is in automatic answer mode when you leave your desk. The tone put out by the AUTOMODEM (the carrier) is *very* loud, and will doubtless get you a few comments if your friends call and the 64 answers! Of course, you must remember to switch to "T" (telephone).

The switch on the right is for half or full duplex. Selecting full duplex prevents you from receiving your own typed-characters back except when echoed by the receiving computer. Some machines you might want to talk to may require half duplex, however, so CBM has included that option. Usually, the software (terminal program) will also allow you to select full or half duplex as well, but switch-selecting is much easier.

In addition to the three switches, there is a CARRIER-DETECT light on the front of the modem. When you are calling another computer and are in the originate mode, you will see this light come on (RED) when your 64 or VIC 20 has made contact with the other machine. If in the answer mode (someone else is calling you), then the light stays on at all times, since your machine is providing the carrier tone. If you autodial another computer and don't get the carrier detect light within a few seconds, usually that means the machine isn't answering (maybe it hasn't paid the bills??) or you got a busy signal. Make sure that your terminal software looks for the carrier-detect, as it makes life much easier for you when you use the autodial functions.

In summation, I feel the AUTOMODEM is a fine product. Commodore has obviously taken pains to produce a good unit, and for that I am grateful. So many new machines and their peripherals are produced on the spur of the moment today with resultant deficiencies, so it's nice to see a piece of equipment done right! Of course the new AUTOMODEM may not be capable of all the "bells and whistles" of some of the higher-priced modems, but for the small cost of the AUTOMODEM, you get plenty! **C**

Commodore 64 Sprite Mover

by Elizabeth Deal

This article is for users of Commodore 64 systems. The program was tested on the PET and in versions one and two of the Commodore 64. The discussion of moving several sprites at a time presupposes your knowledge of the sprite chapters in the user's guide and the *Commodore 64 Programmer's Reference Guide*.

Moving sprites about the Commodore 64 screen is straightforward. The VIC chip keeps track of and rarely forgets the x and y coordinates of the sprite. Sprite coordinate registers are located at hex address VIC=$D000 (53248 decimal). A pair of registers belongs to each sprite, first x then y. One more register is used, hex $D010 (53264) to keep track of a part of the x-coordinate of all eight sprites together.

Specifically, the low byte of the x-coordinate goes in the pairs at $D000, and if x is larger than 255 then a bit is set in the $D010 location. Bits are numbered from right to left, zero to seven, like this:

bit#	7	6	5	4	3	2	1	0
hex value	80	40	20	10	8	4	2	1
decimal	128	64	32	16	8	4	2	1

If sprites three and five have x-coordinates exceeding 255 the pattern is:

bit#	7	6	5	4	3	2	1	0
	0	0	1	0	1	0	0	0

Since this register holds the data for all sprites, we have to carefully look at the register, change only the bit belonging to our sprite and put the value back.

No big deal, really. Let's first do it in BASIC for sprite number three, with the assumption, of course, that all other sprites are live and need protection. The key lines might look like this:

POKEVI+2*3,XAND255 enters low byte of x
POKEVI+16,PEEK(VI+16)AND(255−8) to turn off a bit, and
POKEVI+16,PEEK(VI+16)OR8 to turn it on without affecting other sprites.

This type of code is nicely explained and used in the *Programmer's Reference Guide* in several places, so I won't explain it any further. If you are not understanding what you are now reading it would be a very good idea to review the *Guide*. It is a book of great value.

The Problem

The problem with this code is that it is slooooow. For one or two hot air balloons BASIC is fine. But to move a space vehicle across the screen, the hot air balloon speed just won't do. And if you're defending the world from fast alien beings you may well have to hide instead.

Also, when a sprite crosses over the x=255 line in either direction, everything halts for a while until BASIC copies with the calculations. Finally, a sprite may flicker from one edge of the screen to another during the calculation, a slight annoyance.

A Partial Solution

To remedy all this, we put the works in machine code. The code is the same, the flicker is probably there, but you won't see it happen. Honest.

Usually in this place most authors say, "And machine code, of course, runs several zillion times faster". Well, I cannot say it, because it does not. My luck! It runs much faster than BASIC and the motion is smoother, but it is not lightning speed at all, and the speed depends on the number of live sprites, which is still undesirable.

The reason is that BASIC, now somewhat slowed down by the presence of sprites on the screen, still controls the process. See the loop in lines 490-550 in the DEMO listing. For superfast motion we need to get out of BASIC altogether, or retain BASIC logic but move the sprites using the interrupts. But that's a whole other story.

In the meantime, what you are now getting is a subroutine that can be used as a building block. It is relocatable. It is self-contained. It can become a part of your sprite utilities. And it can be used in an interrupt-driven routine if you do the setup properly. You'll need to reserve some memory for sprite data:

the initial locations, frequency of changing the locations to keep the speed down (timers) and the distances to move. Revector the IRQ routine to point to your code, do JSR MOVER and it will move.

Inputs to the Mover

The values that my routine needs are:

1. Sprite number, 0-7.
2. Direction of motion, coded.
3. How many pixels to move. Be reasonable, but 255 is OK; I'm not checking it.

See the coding protocol in lines 1100-1300 of the PAL listing and an example of use in lines 510-530 of the DEMO program. The direction values in the DEMO program are in a string V$. That should keep us out of trouble with mixing DATA lines.

Note that the direction parameter is closely related to the joystick values, so that if your routine uses a joystick you have little to do but to enter the joystick position.

Typing Instructions

1. Type only what is needed, according to what your computer has. SAVE what you typed before running it, since one tiny little mistake could crash the machine and you'd have to type all over again.
   (1a) If you have an assembler such as PAL64 or the Commodore assembler, you can type (and improve) the source code from line 1000 down to 2080, making the necessary syntax adjustments. I believe that the only difference in the Commodore assembler is that you can only type one instruction per line, so make that change. Other assemblers may need to change some other syntax.
   (1b) If you have SUPERMON64, the simplest thing is to type in lines 2120. Save the code, and you can move it yourself anytime to wherever you please by either an offset load routine of your own making or the SUPERMON's "T" (transfer) command. You'll need to reserve a hunk of safe memory. For that you can follow a pattern in the BASIC loader program, lines 150-240.
   (1c) If you have only BASIC in your 64, you have a tough job of crooked typing ahead. All the DATA lines up front need to be entered. But you get the benefit in that the loader will place the code and protect it from BASIC's harm at the top of the memory. It will check your typing: a checksum error will appear if there are errors. This isn't 100% foolproof, so verify your typing anyway. Otherwise a SYS address will announce itself, such as SYS 12345. If you have SAVEd the DATA lines, you can now type NEW to get rid of the loader, or you can leave it in. It makes no difference.
2. Whichever way you pick, write down the starting address of the code, it may change daily, depending where you put it.
3. Enter this value into the DEMO program in line 620 where you now see illegal code AD= ***.
4. Run the DEMO by saying RUN 450. You'll see a bunch of cute zero-page sprites moving about. If you vary the number of active sprites in line 500 by changing the loop index you'll see a dramatic speed difference in motion.
   (**Note:** *These sprites are actually non-uniform block shapes.)*
5. Type Q to disable the sprites and quit.

Fun Sprites

The sprites used in the DEMO listing are taken from the computer. They are fun to watch. Use of built-in sprites simplifies program writing.

You can do more. You can watch BASIC executing while the sprites are on the screen. A fun thing to do is to use the stop key to kill the program and retain the sprites on the screen. You'll see the displays change. Direct mode print commands and LIST request look pretty good. If you tap keys on the keyboard, this also will show up, so do your own thing and have fun.

Try not to use tape while sprites are on the screen. It won't work. Disk works all right, but if sprites are in motion the motion will halt every once in a while. This is irrelevant to this article, but might be of some use to somebody.

If you do use a stop-key to see the sprites do their thing in direct mode, you can later cleanly disable them. Just type GOTO 570 after the show is over. You will not have to use the drastic measures of the restore key.

Note to PET People

This code can be tested on the PET if you change all references to the VIC chip ($D000) to some other place. When all the arithmetic is right, a final change to the VIC chip address is all that's necessary. **C**

1. *PAL is an assembler written by Brad Templeton distributed by Pro-Line Software Ltd., Canada.*
2. *Commodore assembler is distributed by Commodore dealers.*
3. *SUPERMON64 is Jim Butterfield's monitor, in the public domain.*

Sprite Mover3

```
120 REM-------------------------------
130 REM SPRITE MOVER   ELIZABETH DEAL
140 REM-------------------------------
150 REM BASIC LOADER TO TOP OF MEMORY
160 TM=55:CS=0:READ SP,N$,CV
170 AD=PEEK(TM)+256*PEEK(TM+1)-SP-1
180 V%=AD/256:V=AD-256*V%
190 POKE TM,V:POKE TM+1,V%
200 POKE TM-4,V:POKE TM-3,V%
210 FOR I=0 TO SP:READ V:CS=CS+V
220 POKE AD+I,V:NEXT I
230 IF CS<>CV THEN PRINT"FIX[SPACE]DATA":END
240 PRINT N$":[SPACE]SYS"AD:END
250 DATA  125,"SPRITE[SPACE]MOVER",15816
260 DATA  166,170,165,171, 72,240,115,201
270 DATA   12,176,111,201,  3,240,107,224
280 DATA    8,176,103,138,168, 10,170, 70
290 DATA  171,  8,152, 72,144,  8,164,172
300 DATA  222,  1,208,136,208,250, 70,171
310 DATA  144,  8,164,172,254,  1,208,136
320 DATA  208,250,104,168, 40,240, 67, 56
330 DATA  169,  0, 42,136, 16,252,133,173
340 DATA   45, 16,208,240,  1,200,200, 70
350 DATA  171,144, 12,189,  0,208,229,172
360 DATA  157,  0,208,152,233,  0,168, 70
370 DATA  171,144, 13, 24,165,172,125,  0
380 DATA  208,157,  0,208,152,105,  0,168
390 DATA  152, 41,  1, 74,165,173, 73,255
400 DATA   45, 16,208,144,  2,  5,173,141
410 DATA   16,208,104,133,171, 96
```

```
420 :
430 REM---DEMO SPRITE MOVER----------
440 REM   SEND 8 SPRITES IN 8 DIR
450 GOSUB 600:PRINT CHR$(152);:REM SETUP
455 PRINT"USE[SPACE]Q[SPACE]TO[SPACE]QUIT";
460 POKE VI+21,255          :REM ALL ENABLE
470 Z=35
480 IF PEEK(AD)*PEEK(AD+2)<>166*165 THEN STOP:***
490 FOR I=1 TO Z
495 GET I$:IF I$="Q"GOTO 570
500 FOR J=5 TO 7              :REM 0-7 OK
510 POKE Z1,J                :REM SPRITE#
520 POKE Z2,DR(J+1-(J>=4))   :REM DIR
530 POKE Z3,2                :REM PIXELS
540 IF Z THEN SYS(AD)          :REM MOVE IT
550 NEXT J,I
560 Z=0:GOTO 490:            :REM SILLIES
570 POKE VI+21,0             :REM ALL DISABLE
580 END
590 REM---SETUP
600 V$="060210040008050109":K=0
610 FOR I=1 TO LEN(V$)STEP 2:K=K+1:DR(K)=VAL(MID$(V$,I,2))
    :NEXT I
620 AD= *** :Z1=170:Z2=Z1+1:Z3=Z1+2
630 VI=13*4096      :REM VIC CHIP REG
640 POKE VI+33,0     :REM BLACK BACKGR
650 FOR I=0 TO 7
660 VV=2*I
670 POKE2040+I,I    :REM Z-PAGE ETC SPR
680 POKEVI+VV,220   :REM X-COORD
690 POKEVI+VV+1,130:REM Y-COORD
700 POKEVI+39,15-I :REM ANY COLORS
710 NEXTI
720 POKEVI+23,255   :REM X EXPAND ALL
730 POKEVI+29,255   :REM Y EXPAND ALL
740 POKEVI+28,255   :REM ALL MULTICOLOR
750 POKEVI+37,6     :REM MCOL1
760 POKEVI+38,3     :REM MCOL2
770 POKEVI+16,0     :REM BEGIN ALL X<256
780 RETURN
790 :
1000 REM----------------------------
1010 REM SYS700
1020 .OPT OO
```

```
1040 ;
1050 ;C64 SPRITE MOVER  ELIZABETH DEAL
1060 ;
1070 ;PAL64 SOURCE
1080 ;CODE IS RELOCATABLE ANYPLACE
1090 ;
1100 ;BASIC POKES SPRITE#,DIRECTION,#PIXELS TO MOVE
1110 ;DIRECTION IS A 4-BIT VALUE RLDU
1120 ;SAME AS C64 JOYSTICK WITH BITS FLIPPED
1130 ;
1140 ;" 8421 <-- D        5  1  9
1150 ;" RLDU               \ /
1160 ;"                   4- 0 -8
1170 ;"                    / \
1180 ;"                   6  2 10
1190 ;
1200 ;MOVER CAN BE USED VIA INTERUPT IF TIMING
1210 ;IS DONE IN ANOTHER ROUTINE AND
1220 ;SPRITE#,DIRECTION,DISTANCE ARE
1230 ;SET UP BEFORE CALLING THE MOVER
1240 ;
1250 ;THE BASIC CONNECTION
1260 ;" FORD=1TO9:READ DR(D):NEXTD
1270 ;" DATA 06,02,10,04,00,08,05,01,09
1280 ;" SN=4:D=3:MP=2:REM SPRITE #4,SOUTH-EAST,2PIXELS
1290 ;" POKE170,SN:POKE171,DR(D):POKE173,MP:SYS MOVER
1300 ;
1310 ;---
1320 ;
1330 SN =$AA            ;SPRITE NUMBER 0-7
1340 DIR =SN+1          ;DIRECTION 0-15
1350 PIX =SN+2          ;PIXEL DIST <255
1360 MASK =SN+3         ;HOLDS 2^SN
1370 VIC =$D000         ;VIC CHIP C64
1380 ;
1390 ;---
1400 MOVER =*
1410 LDX SN:LDA DIR:PHA
1420 ;---CHECK SN,DIR PARAMETERS
1430 ;   USER RESPONSIBLE FOR PIX
1440 BEQ EXIT1          ;LOOOONG JUMP
1450 CMP #$C:BCS EXIT1
1460 CMP #3:BEQ EXIT1
1470 CPX #8:BCS EXIT1
1480 ;---PARAMETERS OK
```

```
1490 TXA:TAY:ASL:TAX  ;X=2*SN  Y=SN
1500 ;---DO Y-COORDINATE
1510 LSR DIR:PHP
1520 TYA:PHA
1530 BCC DOWN
1540 LDY PIX
1550 U  DEC VIC+1,X   ;MOVE UP
1560 DEY:BNE U
1570 DOWN LSR DIR
1580 BCC YDONE
1590 LDY PIX
1600 D  INC VIC+1,X   ;MOVE DOWN
1610 DEY:BNE D
1620 YDONE =*
1630 PLA:TAY:PLP
1640 ;---DO X-COORDINATE
1650 HORIZ =*
1660 BEQ EXIT1        ;NO HORIZONTAL
1670 SEC              ;DO 2^SPR#
1680 LDA #0           ;    (THIS IS
1690 SETBIT ROL       ;    SLOW...BUT
1700 DEY              ;    RELOCATABLE)
1710 BPL SETBIT       ;Y=$FF
1720 STA MASK         ;KEEP RESULT
1730 AND VIC+$10      ;TEST IF HORIZ>255 NOW
1740 BEQ ZERO
1750 INY              ;YES
1760 ZERO INY         ;NO   Y=HI BYTE X-COORD (0 OR 1)
1770 LSR DIR          ;--TRY LEFT DIR
1780 BCC RIGHT
1790 LDA VIC,X
1800 SBC PIX
1810 STA VIC,X        ;HORIZ-PIX, LOW BYTE
1820 TYA:SBC #0:TAY   ;           HI  BYTE
1830 ;
1840 RIGHT LSR DIR    ;--TRY RIGHT DIR
1850 BCC MSBX
1860 CLC:LDA PIX
1870 ADC VIC,X
1880 STA VIC,X        ;HORIZ+PIX, LOW BYTE
1890 TYA:ADC #0:TAY   ;           HI  BYTE
1900 ;
1910 MSBX =*          ;--FIX HI-BYTE IN VIC+16, LIMIT 1
1920 TYA:AND #1:LSR   ;SET CARRY BIT
1930 LDA MASK
```

```
1940 EOR #$FF          ;FLIP 2^SN MASK
1950 AND VIC+$10       ;TURN OFF BIT SN
1960 BCC PUTBACK
1970 ORA MASK          ;ON IF NEEDED
1980 PUTBACK STA VIC+$10
1990 ;
2000 EXIT1  PLA:STA DIR ;MAY NEED IT
2010 RTS
2020 ;
2030 ;STATUS - A,Y,FLAGS UNDEFINED
2040 ;         X=2*SN
2050 ;         SN,DIR,PIX UNCHANGED
2060 ;         VIC X,Y SN POSITIONS UPDATED
2070 .END
2080 END
2090 REM------------------------------
2100 REM MEMORY DUMP TO COPY ANYPLACE
2110 REM USING SUPERMON64
2120 SYS8
2130 "
2140 B*
2150 "  PC  SR AC XR YR SP
2160 .;0008 F0 00 08 A8 F6
2170 .M 7000 707E
2180 .:7000 A6 AA A5 AB 48 F0 73 C9
2190 .:7008 0C B0 6F C9 03 F0 6B E0
2200 .:7010 08 B0 67 8A A8 0A AA 46
2210 .:7018 AB 08 98 48 90 08 A4 AC
2220 .:7020 DE 01 D0 88 D0 FA 46 AB
2230 .:7028 90 08 A4 AC FE 01 D0 88
2240 .:7030 D0 FA 68 A8 28 F0 43 38
2250 .:7038 A9 00 2A 88 10 FC 85 AD
2260 .:7040 2D 10 D0 F0 01 C8 C8 46
2270 .:7048 AB 90 0C BD 00 D0 E5 AC
2280 .:7050 9D 00 D0 98 E9 00 A8 46
2290 .:7058 AB 90 0D 18 A5 AC 7D 00
2300 .:7060 D0 9D 00 D0 98 69 00 A8
2310 .:7068 98 29 01 4A A5 AD 49 FF
2320 .:7070 2D 10 D0 90 02 05 AD 8D
2330 .:7078 10 D0 68 85 AB 60 68 85
2340 .S"SPRITE MOVER",08,7000,707E DISK
2350 .S"SPRITE MOVER",01,7000,707E TAPE
2360 .X
2370 READY.
2380 ;--------------------------------
```

Go Directly to XY!

by David Bull

—Being a compendium of kludges, patches and subroutines all aimed at making up for the lack of a PRINT AT statement in Commodore BASIC. Some are short, some long, some quick, some slow. Perhaps one is right for your particular application. Routines are written for CBM/PET computers with 4.0 BASIC, but the technique applies to all Commodore computers.

At some point in our programming adventures, every one of us using Commodore products may have been frustrated by the missing PRINT AT statement in CBM BASIC. This extension to the normal PRINT statement allows the programmer to specify the exact screen location at which data is to appear. The beginner may not notice its absence, as he is usually content using his "glass teletype", but for any non-trivial data display using PRINT statements, the AT is sorely missed.

Fortunately, PRINT AT is easily simulated. What is not quite so easy is finding the optimal solution. For one person the most important consideration is speed, whereas someone else may be more concerned with memory limitations.

Follow me, as I explore some of the alternative techniques.

In order to find the "best" solution, we will need some way to measure our attempts. I will use the program in Listing 1 as a test bed. It simply provides a framework in which we can insert each subroutine as we develop it, and then exercise that routine by making many calls to it. For a rough-and-ready reckoning of memory usage we will compare a FRE(0) value obtained after each test, with the value given by the test bed alone.

Note: throughout this article I will follow the convention—x = screen row, and y = screen column.

On with the show!

Let's start with everybody's favorite:

—home cursor
—print 'x-1' cursor down characters
—print 'y-1' cursor right characters
—print the data

Here's the simple subroutine to do this:

```
10 PRINT  CHR$(19);
12 IF X<2 THEN 16
14 FOR Z=1 TO X-1:PRINT CHR$(17)
   ;:NEXT Z
16 IF Y<2 THEN 20
18 FOR Z=1 TO Y-1:PRINT CHR$(29)
   ;:NEXT Z
20 RETURN
```

Most versions I have seen in this routine forget the tests for skipping the loops, which otherwise are always executed, making it impossible to ever print on the top row or in the first column.

A typical call to use all our subroutines:

```
XXXX x=12 : y=24 : gosub10 : print
     "Hi mom!"
```

Time to run test/a . . . . . 252.64 seconds
Bytes used over and above the testbed 115

Hmm... nice 'n easy, but not too fast. Our operator will get pretty bored waiting for a screenful of data to be positioned and printed at this speed. How about packing all that printing into one statement?

Somewhere at the beginning of our program we will need the initialization:

```
2 X$="":FOR INDEX=1 TO 24:X$=X$
  +CHR$(17):NEXT INDEX
3 Y$="":FOR INDEX=1 TO 79:Y$=Y$
  +CHR$(29):NEXT INDEX
```

To position the cursor the subroutine line will read:

```
10 PRINT CHR$(19);MID$(X$,1,X-1)
   ;MID$(Y$,1,Y-1);:RETURN
```

We use MID$ rather than LEFT$ because some older Commodore BASICs do not accept zero as a parameter for LEFT$. Using MID$ will avoid any problems.

Simple and short. How does it stack up?

Time to run test/b . . . . . 52.13 seconds
Bytes used over and above the testbed 247

Amazing! Six times faster. But... it cost us a lot of bytes. What about coding it in machine language? Can we improve on this speed, and perhaps save some memory?

```
., Ø3CA A9 13    LDA #$13
., Ø3CC 2Ø 66 F2 JSR $F266
., Ø3CF A9 11    LDA #$11
., Ø3D1 AE EC Ø3 LDX $Ø3EC
., Ø3D4 CA       DEX
., Ø3D5 FØ Ø6    BEQ $Ø3DD
., Ø3D7 2Ø 66 F2 JST $F266
., Ø3DA CA       DEX
., Ø3DB DØ FA    BNE $Ø3D7
., Ø3DD A9 1D    LDA #$1D
., Ø3DF AE ED Ø3 LDX $Ø3ED
., Ø3E2 CA       DEX
., Ø3E3 FØ Ø6    BEQ $Ø3EB
., Ø3E5 2Ø 66 F2 JSR $F266
., Ø3E8 CA       DEX
., Ø3E9 DØ FA    BNE $Ø3E5
., Ø3EB 6Ø       RTS
., Ø3EC XX       ???
., Ø3ED XX       ???
```

The memory locations given here for storage of this routine are in the second cassette buffer. The $13 represents the "home cursor" character, the $11 is the "cursor down", and the $1D is the "cursor right". The jump to the ROM routine at $F266 takes these characters and puts them on the screen. The x value must be POKEd into $03EC and the y value into $03ED before the routine is called. Once the machine language is in place in our machine, the command to use it is:

```
 0 ON Z GOTO 100
 1 Z=1:LOAD"0:XYLOCATE",08:REM
   XYLOCATE IS THE MACHINE LAN
   G. PORTION
10 POKE 1004,(X):POKE 1005,(Y)
   :SYS 970:RETURN
```

This routine assumes an intelligent use. It will not give coherent results if either x or y is given a "far-out" value.

Time to run test/c . . . . . 49.79 seconds
Bytes used over and above the testbed 52

Good... a bit faster and much more memory efficient (as the machine language in the cassette buffer is invisible to BASIC). But, come to think of it, how did that ROM routine keep track of where it was while it was putting things on the screen? Poring over some memory maps shows something interesting at locations 196, 197, and 198 (that's for BASIC 4.0 and upgrade ROMS. Use 224, 225, and 226 for original ROMS). 196/7 is the pointer to the screen RAM line. 198 is the position of the cursor on above line. Let's experiment a little bit in direct mode.

```
poke 196,0 : poke 197,128 : poke
     198,28 : print "Hi mom!"
```

This statement should put the message about half-way along the top of the screen. How does it work?

The machine expects that the values placed in 196/7 will represent one of the memory locations in the video RAM section of memory, as shown in Figure 1.

The number in 198 is simply an offset, which will be added to this value to adjust the print position across the screen to the right.

So, how do we convert our x and y coordinates to these values? This subroutine should do it:

```
10 X=((X-1)*80)+32768
12 MSB=INT(X/256)
14 LSB=X-(256*MSB)
16 POKE 196,(LSB):POKE 197,(MSB)
   :POKE 198,(Y-1)
18 RETURN
```

Line 10 converts x into the appropriate value from Figure 1. Lines 12 and 14 split that value into the two-byte format required by the 6502 processor. Voila! When a PRINT statement is executed the ROM routines will find the POKEd values and use them to position the message.

Time to run test/d . . . . . 73.58 seconds
Bytes used over and above the testbed 124

Hmm... back to the drawing board... all that arithmetic must take time. What if we put *this* one into machine language?

```
., Ø3CA A9 ØØ    LDA #$ØØ
., Ø3CC 85 C4     STA $C4
., Ø3CE A9 8Ø     LDA #$8Ø
., Ø3DØ 85 C5     STA $C5
```

```
., Ø3D2 A2 ØØ     LDX #$ØØ
., Ø3D4 FØ 1Ø     BEO $Ø3E6
., Ø3D6 18        CLC
., Ø3D7 A5 C4     LDA $C4
., Ø3D9 69 5Ø     ADC #$5Ø
., Ø3DB 85 C4     STA $C4
., Ø3DD A5 C5     LDA $C5
., Ø3DF 69 ØØ     ADC #$ØØ
., Ø3E1 85 C5     STA $C5
., Ø3E3 CA        DEX
., Ø3E4 DØ FØ     BNE $Ø3D6
., Ø3E6 6Ø        RTS
```

This routine takes our x value (which we must POKE into $03D3 (decimal 979) before calling the routine) and uses it as a counter:

—put 32768 in locations 196/7
—if x = 0 then leave 196/7 alone and return
* —add '80' to value stored in locations 196/7
—decrement x (x = x-1)
—if x = 0 then quit (return)
—else loop back to *

So, with the address of the appropriate row in 196/7, and with the y offset POKEd into 198, print statements will appear where desired. The sub-routine in BASIC:

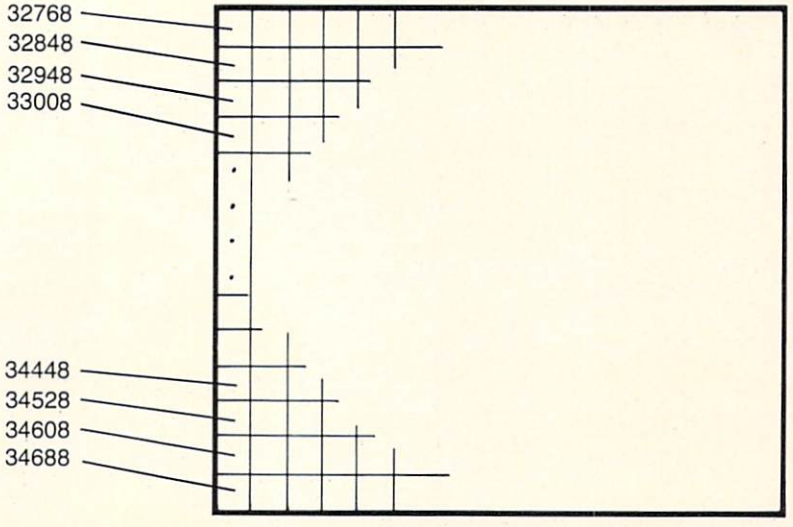


Figure 1.

```
0 ON Z GOTO 100
1 Z=1:LOAD"0:XYLOCATE/B",08:REM
  XYLOCATE/B IS THE MACH. L
  ANG. PORTION
10 POKE 979,(X-1):POKE 198,(Y-1)
   :SYS 970:RETURN
```

Time to run test/e ..... 43.68 seconds
Bytes used over and above the testbed 56

Now we're cooking with gas! But wait! Why are we fooling around like this? Think for a minute. How does the computer know where to print the ***COMMODORE BASIC*** message when it is first turned on? Surely, buried in all that ROM somewhere, there must be a routine for doing exactly what we want—preparing the cursor for printing. Back to the memory maps... aha! there it is! "Screen initialization"—$E000. It turns out that deep in ROM there is a two-part table containing all those numbers from Figure 1. Our x value can be used as an offset to read the appropriate number from the table, which is then stored in those same locations that we used in test/d.

```
., EØ67 A6 D8    LDX $D8
., EØ69 4C 6F EØ JMP $EØ6F
```

```
., EØ6C XX XX    ???
., EØ6E XX       ???
., EØ6F BD 55 E7 LDA $E755,X
., EØ72 85 C4    STA $C4
., EØ74 BD 6E E7 LDA $E76E,X
., EØ77 85 C5    STA $C5
., EØ79 6Ø       RTS
```

So:

```
10 POKE 216,(X-1):POKE 198,
   (Y-1):SYS 57447:RETURN
```

These are the values for 80-column BASIC 4.0. Upgrade ROMs should use the same POKEs as this, while original ROMs will use 245 for x, and 226 for y. I don't have access to other ROM machines (or a 40-column 4.0) so I can't give you the SYS values for them. Perhaps someone will do a little digging in their machines and send them in.

Time to run test/f ..... 45.66 seconds
Bytes used over and above the testbed 40

The clear winner... extremely fast and memory efficient (thanks to that table).

So, PRINT AT was in there all the time! Ah well, it was fun anyway, and *most* instructive. **C**

Testbed

```
5 GOTO 100 : REM START OF PROGRAM
6 :
10 REM
12 RETURN
50 :
100 PRINT CHR$(147);
110 T=TI
120 FOR INDEX = 1 TO 500
130 : X=INT(12*RND(0))+1
140 : Y=INT(80*RND(0))+1
150 : GOSUB 10 : PRINT "*";
155 : X=12:Y=13:GOSUB 10:PRINT
    INDEX"[SPACE]STARS[SPACE]
    PRINTED[SPACE]";
160 NEXT INDEX
170 :
200 X=12:Y=13:GOSUB 10:PRINT "
    500[SPACE]STARS[SPACE]PRINTED
    [SPACE](GOTOXY[SPACE]1000[SPA
    CE]TIMES)[SPACE]IN[SPACE]";
220 PRINT INT(((TI-T)/60)*100)
    /100;"[SPACE]SECONDS"
230 END
```

user groups

User Group Listing

ALABAMA

Huntsville PET Users Club
9002 Berclair Road
Huntsville, AL 35802
Contact: Hal Carey
Meetings: every 2nd Thursday

Riverchase Commodore Users Group
617 Grove St.
Birmingham, AL 35209
(205) 988-1078
Ken Browning

Wiregrass Micro-Computer Society
Commodore SIG
109 Key Bend Rd.
Enterprise, AL 36330
(205) 347-7564
Bill Brown

ALASKA

COMPOOH-T
c/o Box 118
Old Harbor, AK 99643
(907) 286-2213

ARIZONA

VIC Users Group
2612 E. Covina
Mesa, AZ 85203
Contact: Paul Muffuletto

Catalina Commodore Computer Club
2012 Avenida Guillermo
Tucson, AZ 85710
(602) 296-6766
George Pope
1st Tues. 7:30 p.m.

Central Arizona PET People
842 W. Calle del Norte
Chandler, AZ 85224
(602) 899-3622
Roy Schahrer

ACUG
c/o Home Computer Service
2028 W. Camelback Rd.
Phoenix, AZ 85015
(602) 249-1186
Dan Deacon
First Wed. of month

West Mesa VIC
2351 S. Standage
Mesa, AZ 85202
Kenneth S. Epstein

Arizona VIC 20-64 Users Club
232 W. 9th Place North
Mesa, AZ 85201
Donald Kipp

Arizona VIC & 64 Users
904 W. Marlboro Circle
Chandler, AZ 85224
(602) 963-6149
Tom Monson

ARKANSAS

Commodore/PET Users Club
Conway Middle School
Davis Street
Conway, AR 72032
Contact: Geneva Bowlin

Booneville 64 Club
c/o A. R. Hederich
Elementary School
401 W. 5th St.
Booneville, AR 72927
Mary Taff

The Siloam Commodore Computer Club
P.O. Box 88
Siloam Springs, AR 72761
(501) 524-5624
Ken Emanualson

Russellville Commodore User Group
401 S. Arlington Dr.
Russellville, AR 72801
(501) 967-1868
Bob Brazeal

CALIFORNIA

SCPUG Southern California
PET Users Group
c/o Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: First Tuesday of each month

California VIC Users Group
c/o Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: Second Tues. of each month

Valley Computer Club
1913 Booth Road
Ceres, CA 95307

PUG of Silicon Valley
22355 Rancho Ventura Road
Cupertino, CA 95014

Lincoln Computer Club
750 E. Yosemite
Manteca, CA 95336
John Fung, Advisor

PET on the Air
525 Crestlake Drive
San Francisco, CA 94132
Max J. Babin, Secretary

PALS (Pets Around)
Livermore Society
886 South K
Livermore, CA 94550
(415) 449-1084
Every third Wednesday
7:30 p.m.
Contact: J. Johnson

SPHINX
7615 Leviston Ave.
El Cerrito, CA 94530
(415) 527-9286
Bill MacCracken

San Diego PUG
c/o D. Costarakis
3562 Union Street
(714) 235-7626
7 a.m.-4 p.m.

Walnut Creek PET
Users Club
1815 Ygnacio Valley Road
Walnut Creek, CA 94596

Jurupa Wizards
8700 Galena St.
Riverside, CA 92509
781-1731
Walter J. Scott

The Commodore Connection
2301 Mission St.
Santa Cruz, CA 95060
(408) 425-8054
Bud Massey

San Fernando Valley
Commodore Users Group
21208 Nashville
Chatsworth, CA 91311
(213) 709-4736
Tom Lynch
2nd Wed. 7:30

VACUUM
277 E. 10th Ave.
Chico, CA 95926
(916) 891-8085
Mike Casella
2nd Monday of month

VIC 20 Users Group
2791 McBride Ln. ;121
Santa Rosa, CA
(707) 575-9836
Tyson Verse

South Bay Commodore Users Group
1402 W. 218th St.
Torrance, CA 90501
Contact: Earl Evans

Slo VIC 20/64 Computer Club
1766 9th St.
Los Osos, CA

The Diamond Bar R.O.P. Users Club
2644 Amelgado
Haciendo Hgts., CA 91745
(213) 333-2645
Don McIntosh

Commodore Interest Association
c/o Computer Data
14660 La Paz Dr.
Victorville, CA 92392
Mark Finley

Fairfield VIC 20 Club
1336 McKinley St.
Fairfield, CA 94533
(707) 427-0143
Al Brewer
1st & 3rd Tues. at 7 p.m.

Computer Barn Computer Club
319 Main St.
Suite #2
Salinas, CA 93901
757-0788
S. Mark Vanderbilt

Humboldt Commodore Group
P.O. Box 570
Arcata, CA 95521
R. Turner

Napa Valley Commodore
Computer Club
c/o Liberty Computerware
2680 Jefferson St.
Napa, CA 94558
(707) 252-6281
Mick Winter
1st & 3rd Mon. of month

S.D. East County C-64 User Group
6353 Lake Apopka Place
San Diego, CA 92119
(619) 698-7814
Linda Schwartz

Commodore Users Group
4237 Pulmeria Ct.
Santa Maria, CA 93455
(805) 937-4174
Gilbert Vela

Bay Area Home Computer Asso.
Walnut Creek Group
1406 N. Broadway at Cypress
Walnut Creek, CA 94596
Wil Cossel
Sat. 11 a.m. to 3 p.m.

Amateurs and Artesians Computing
P.O. Box 682
Cobb, CA 95426

Manteca VIC 20 Users Organization
429 N. Main St.
Manteca, CA 95336
Gene Rong

Pomona Valley Vic Users Group
1401 W. 9th, #77
Pomona, CA 91766
(714) 620-8889
Mark Joerger
1st & 3rd Wed. of month 7 p.m.

20/64 Users Group
P.O. Box 18473
San Jose, CA 95158
Don Cracraft
1st Sunday, 6 p.m., Mercury Sav

VIC TORII-The VIC 20 Users Group
PSC #1, Box 23467
APO San Francisco, CA 96230
Wesley Clark

The Valley Computer Club
2006 Magnolia Blvd.
Burbank, CA 91506
1st Wed. 7 p.m.

The Commodore Tech. Users
of Orange Co.
P.O. Box 1497
Costa Mesa, CA 92626
(714) 731-5195
Roger Fisher

VIC 20 Software Exchange Club
10530 Sky Circle
Grass Valley, CA 95945
Daniel Upton

C-64 West Orange County Users Group
P.O. Box 1457
Huntington Beach, CA 92647
(714) 842-4484
Philip Putman
2nd & 4th Tues. of month

Antelope Valley Commodore Users Group
POB 4436
Lancaster, CA 93539
(805) 942-2626
James Haner
1st Saturday

Diablo Valley Commodore Users Group
762 Ruth Dr.
Pleasant Hill, CA 94523
(415) 671-0145
Ben Braver
2nd & 4th Thurs. 7:30 p.m.

Commodore Connection
11652 Valverde Ave.
Riverside, CA 92505
(714) 689-7447
Tony Alvarez

CA. Area Commodore Terminal
Users Society
C.A.C.T.U.S.
P.O. Box 1277
Alta Loma, CA 91701
Darrell Hall

user groups

20/64
P.O. Box 18473
San Jose, CA 95158
(408) 978-0546
1st Sun. of month (6-9 p.m.)

8120 Sundance Dr.
Orangevale, CA 95662
(916) 969-2028
Robyn Graves

COLORADO

VICKIMPET Users Group
4 Waring Lane, Greenwood Village
Littleton, CO 80121
Contact: Louis Roehrs

Colorado Commodore Computer Club
2187 S. Golden Ct.
Denver, CO 80227
986-0577
Jack Moss
Meet: 2nd Wed.

CONNECTICUT

John F. Garbarino
Skiff Lane Masons Island
Mystic, CT 06355
(203) 536-9789

Commodore User Club
Wethersfield High School
411 Wolcott Hill Road
Wethersfield, CT 06109
Contact: Daniel G. Spaneas

VIC Users Club
c/o Edward Barszczewski
22 Tunxis Road
West Hartford, CT 06107

New London County Commodore Club
Doolittle Road
Preston, CT 06360
Contact: Dr. Walter Doolittle

FLORIDA

Jacksonville Area PET Society
401 Monument Road, #177
Jacksonville, FL 32211

Richard Prestien
6278 SW 14th Street
Miami, FL 33144

South Florida PET Users Group
Dave Young
7170 S.W. 11th
West Hollywood, FL 33023
(305) 987-6982

PETs and Friends
129 NE 44 St.
Miami, FL 33137
Richard Plumer

Sun Coast VICs
P.O. Box 1042
Indian Rocks Beach, FL 33535
Mark Weddell

Bay Commodore Users Group
c/o Gulf Coast Computer Exchange
241 N. Tyndall Pkwy.
P.O. Box 6215
Panama City, FL 32401
(904) 785-6441
Richard Scofield

Gainesville Commodore Users Club
3604-20A SW 31st Dr.
Gainesville, FL 32608
Louis Wallace

64 Users Group
P.O. Box 561689
Miami, FL 33156
(305) 274-3501
Eydie Sloane

Brandon Users Group
108 Anglewood Dr.
Brandon, FL 33511
(813) 685-5138
Paul Daugherty

Brandon Commodore Users Group
414 E. Lumsden Rd.
Brandon, FL 33511

Gainesville Commodore Users Group
Santa Fe Community College
Gainesville, FL 32602
James E. Birdsell

Commodore Computer Club
P.O. Box 21138
St. Petersburg, FL 33742

Commodore Users Group
545 E. Park Ave.
Apt. #2
Tallahassee, FL 32301
(904) 224-6286
Jim Neill

The Commodore Connection
P.O. Box 6684
West Palm Beach, FL 33405

El Shift OH
P.O. Box 548
Cocoa, FL 32922
Mike Schnoke
Sat. mornings/every 4 to 6 weeks

Miami 20/64
12911 S.W. 49th St.
Miami, FL 33175
(305) 226-1185

Tampa Bay Commodore Computer Club
10208 N. 30th St.
Tampa, FL 33612
(813) 977-0877

GEORGIA

VIC Educators Users Group
Cherokee County Schools
110 Academy St.
Canton, GA 30114
Dr. Al Evans

Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

VIC-tims
P.O. Box 467052
Atlanta, GA 30346
(404) 922-7088
Eric Ellison

Golden Isles Commodore Users Club
Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

HAWAII

Commodore Users Group of Honolulu
c/o PSH
824 Bannister St.
Honolulu, HI
(808) 848-2088
3rd Fri. every month

20/64 Hawaii
P.O. Box 966
Kailua, HI 96734
Wes Goodpaster

Commodore Users Group of Honolulu
1626 Wilder #701
Honolulu, HI 96822
(808) 848-2088
Jay Calvin (808) 944-9380

IDAHO

GHS Computer Club
c/o Grangeville High School
910 S. D St.
Grangeville, ID 83530
Don Kissinger

S.R.H.S. Computer Club
c/o Salmon River H.S.
Riggins, ID 83549
Barney Foster

Commodore Users
548 E. Center
Pocatello, ID 83201
(208) 233-0670
Leroy Jones

Eagle Rock Commodore Users Group
900 S. Emerson
Idaho Falls, ID 83401
Nancy J. Picker

ILLINOIS

Shelly Wernikoff
2731 N. Milwaukee Avenue
Chicago, IL 60647

VIC 20/64 Users Support Group
c/o David R. Tarvin
114 S. Clark Street
Pana, IL 62557
(217) 562-4568

Central Illinois PET User Group
635 Maple
Mt. Zion, IL 62549
(217) 864-5320
Contact: Jim Oldfield

ASM/TED User Group
200 S. Century
Rantoul, IL 61866
(217) 893-4577
Contact: Brant Anderson

PET VIC Club (PVC)
40 S. Lincoln
Mundelein, IL 60060
Contact: Paul Schmidt, President

Rockford Area PET Users Group
1608 Benton Street
Rockford, IL 61107

Commodore Users Club
1707 East Main St.
Olney, IL 62450
Contact: David E. Lawless

VIC Chicago Club
3822 N. Bell Ave.
Chicago, IL 60618
John L. Rosengarten

Chicago Commodore 64 Users & Exchange Group
P.O. Box 14233
Chicago, IL 60614
Jim Robinson

Fox Valley PET Users Group
833 Willow St.
Lake in the Hills, IL 60102
(312) 658-7321
Art DeKneef

The Commodore 64 Users Group
P.O. Box 572
Glen Ellyn, IL 60137
(312) 790-4320
Gus Pagnotta

RAP 64/VIC Regional Assoc. of Programmers
10721 S. Lamon
Oak Lawn, IL 60453
Bob Hughes

The Kankakee Hackers
RR #1, Box 279
St. Anne, IL 60964
(815) 933-4407
Rich Westerman

WIPUG
Rt. 5, Box 75
Quincy, IL 62301
(217) 656-3671
Edward Mills

INDIANA

PET/64 Users
10136 E. 96th St.
Indianapolis, IN 46256
(317) 842-6353
Jerry Brinson

Cardinal Sales
6225 Coffman Road
Indianapolis, IN 46268
(317) 298-9650
Contact: Carol Wheeler

CHUG (Commodore Hardware Users Group)
12104 Meadow Lane
Oaklandon, IN 46236
Contact: Ted Powell

VIC Indy Club
P.O. Box 11543
Indianapolis, IN 46201
(317) 898-8023
Ken Ralston

Northern Indiana Commodore Enthusiasts
927 S. 26th St.
South Bend, IN 46615
Eric R. Bean

Commodore Users Group
1020 Michigan Ave.
Logansport, IN 46947
(219) 722-5205
Mark Bender

Computer Workshop VIC 20/64 Club
282 S. 600 W.
Hebron, IN 46341
(219) 988-4535
Mary O'Bringer

The National Science Clubs of America Commodore Users Division

7704 Taft St.
Merrillville, IN 46410
Brian Lepley or Tom Vlasic

East Central Indiana VIC User Group
Rural Route #2
Portland, IN 47371
Stephen Erwin

National VIC 20 Program Exchange
102 Hickory Court
Portland, IN 47371
(219) 726-4202
Stephen Erwin

Commodore Computer Club
3814 Terra Trace
Evansville, IN 47711
(812) 477-0739
John Patrick, President

Commodore 64 Users Group
912 South Brown Ave.
Terre Haute, IN 47803
(812) 234-5099
Dennis Graham

IOWA

Commodore User Group
114 8th St.
Ames, IA 50010

Quad City Commodore Club
1721 Grant St.
Bettendorf, IA 52722
(319) 355-2641
John Yigas

Commodore Users Group
965 2nd St.
Marion, IA 52302
(319) 377-5506
Vern Rotert
3rd Sun. of month

Siouxland Commodore Club
2700 Sheridan St.
Sioux City, IA 51104
(712) 258-7903
Gary Johnson
1st & 3rd Monday of month

421 W. 6th St.
Waterloo, IA 50702
(319) 232-1062
Frederick Volker

Commodore Computer Users
Group of Iowa
Box 3140
Des Moines, IA 50316
(515) 263-0963 or (515) 287-1378
Laura Miller

KANSAS

Wichita Area PET
Users Group
2231 Bullinger
Wichita, KS 67204
(316) 838-0518
Contact: Mel Zandler

Kansas Commodore
Computer Club
101 S. Burch
Olathe, KS 66061
Contact: Paul B. Howard

Commodore Users Group
6050 S. 183 St. West
Viola, KS 67149
Walter Lounsbery

Walnut Valley Commodore User Group
1003 S. 2nd St.
Arkansas City, KS 67005
Bob Morris

KENTUCKY

VIC Connection
1010 S. Elm
Henderson, KY 42420
Jim Kemp

Louisville Users of Commodore KY.
(LUCKY)
c/o Computer Showroom
1247 Hurstbourne
Louisville, KY 40222
2nd Monday

LOUISIANA

Franklin Parish Computer
Club
#3 Fair Ave.
Winnisboro, LA 71295
James D. Mays, Sr.

NOVA
917 Gordon St.
New Orleans, LA 70117
(504) 948-7643
Kenneth McGruder, Sr.

VIC 20 Users Group
5064 Bowdon St.
Marrero, LA 70072
(504) 341-5305
Wayne D. Lowery, R.N.

64-Club News
5551 Corporate Blvd.
Suite 3L
Baton Rouge, LA 70808
(504) 766-7408
Tom Parsons
3rd Tues. of month at CWA

Commodore Users Group of Oachita
P.O. Box 175
Swaric, LA 71281
(318) 343-8044
Beckie Walker

MARYLAND

Assoc. of Personal
Computer Users
5014 Rodman Road
Bethesda, MD 20016

Blue TUSK
700 East Joppa Road
Baltimore, MD 21204
Contact: Jim Hauff

House of Commodore
8835 Satyr Hill Road
Baltimore, MD 21234
Contact: Ernest J. Fischer

Long Lines Computer Club
323 N. Charles St., Rm. 201
Baltimore, MD 21201
Gene Moff

VIC & 64 Users Group
The Boyds Connection
21000 Clarksburg Rd.
Boyds, MD 20841
(301) 428-3174
Tom DeReggi

VIC 20 Users Group
23 Coventry Lane
Hagerstown, MD 21740
Joseph Rutkowski

Hagerstown Users Group
1201-B Marshall St.
Hagerstown, MD 21740
(301) 790-0968
Greg Stewart
1st & 3rd Friday of month 6:30 p.m.

Rockville VIC/64 Users Group
5112 Parklawn Terrace
Apt. #103
Rockville, MD 20853
(301) 231-7823
Tom Pounds

The Compucats' Commodore
Computer Club
680 W. Bel Air Ave.
Aberdeen, MD 21001
(301) 272-0472
Betty Jane Schueler

Westinghouse BWI
Commodore User Group
Attn: L. Barron Mail Stop 5156
P.O. Box 1693
Baltimore, MD 21203

MASSACHUSETTS

Eastern Massachusetts
VIC Users Group
c/o Frank Ordway
7 Flagg Road
Marlboro, MA 02173

VIC Users Group
c/o Ilene Hoffman-Sholar
193 Garden St.
Needham, MA 02192

Commodore Users Club
Stoughton High School
Stoughton, MA 02072
Contact: Mike Lennon

Berkshire PET Lovers
CBM Users Group
Taconic High
Pittsfield, MA 01201

The Boston Computer
Society
Three Center Plaza
Boston, MA 02108
(617) 367-8080
Mary E. McCann

Masspet Commodore Users Group
P.O. Box 307
East Taunton, MA 02718
David Rogers

Raytheon Commodore Users Group
Raytheon Company
Hartwell Rd. GRA-6
Bedford, MA 01730
John Rudy

Commodore 64 Users
Group of The Berkshires
184 Highland Ave.
Pittsfield, MA 01201
Ed Rucinski

VIC Interface Club
48 Van Cliff Ave.
Brockton, MA 02401
Bernie Robichaud

Cape Cod 64 Users Group
358 Forrest Rd.
S. Yarmouth, MA 02664
1 (800) 225-7136
Jim Close
(In MA. call) 1 (800) 352-7787

MICHIGAN

David Liem
14361 Warwick Street
Detroit, MI 48223

VIC Users Club
University of Michigan
School of Public Health
Ann Arbor, MI 48109
Contact: John Gannon

Commodore User Club
32303 Columbus Drive
Warren, MI 48093
Contact: Robert Steinbrecher

Commodore Users Group
c/o Family Computer
3947 W. 12 Mile Rd.
Berkley, MI 48072

VIC for Business
6027 Orchard Ct.
Lansing, MI 48910
Mike Marotta

South Computer Club
South Jr. High School
45201 Owen
Belleville, MI 48111
Ronald Ruppert

Commodore Users Group
c/o Eaton Rapids Medical Clinic
101 Spicerville Hwy.
Eaton Rapids, MI 48827
Albert Meinke III, M.D.

South East Michigan Pet
Users Group
Box 214
Farmington, MI 48024
Norm Eisenberg

Commodore Computer Club
4106 Eastman Rd.
Midland, MI 48640
(517) 835-5130
John Walley
9:30 p.m. Sept/May

VIC, 64, PET Users Group
8439 Arlis Rd.
Union Lake, MI 48085
363-8539
Bert Searing

VIC Commodore User Club
486 Michigan Ave.
Mariesville, MI 48040
(313) 364-6804
M. Gauthier

ComputerTowne
35171 Grand River
Farmington, MI 48024
(313) 471-4216

Ann Arbor Commodore Users Group
Ann Arbor, MI 48103
(313) 994-4751
Art Shaw
3rd Tues. 7:30-10:00

DAB Computer Club
P.O. Box 542
Watervliet, MI 49098
(616) 463-5457
Dennis Burlingham

West Michigan Commodores
c/o R. Taber
1952 Cleveland Ave., S.W.
Wyoming, MI 49509
(616) 458-9724
Gene Traas

user groups

MINNESOTA

MUPET (Minnesota Users of PET)
P.O. Box 179
Annandale, MN 55302
c/o Jon T. Minerich

Twin Cities Commodore Computer Club
6623 Ives Lane
Maple Grove, MN 55369
(612) 424-2425
Contact: Rollie Schmidt

MISSISSIPPI

Commodore Biloxi User Group (ComBUG)
Universal Computer Services
3002 Hwy. 90 East
Ocean Springs, MS 39564
(601) 875-1173
John Lassen

MISSOURI

KCPUG
5214 Blue Ridge Boulevard
Kansas City, MO 64133
Contact: Rick West
(816) 356-2382

Commodore User Group of St. Louis
Box 6653
St. Louis, MO 63125-0653
Dan Weidman, New Members
1541 Swallowtail Dr.
St. Louis, MO

VIC INFONET
P.O. Box 1069
Branson, MO 65616
(417) 334-6099
Jory Sherman

Worth County PET Users Group
Grant City, MO
(816) 564-3551
David Hardy

Mid-Missouri Commodore Club
1804 Vandiver Dr.
Columbia, MO 65201
(314) 474-4511
Phil Bishop

Joplin Commodore Computers Users Group
422 S. Florida Ave.
Joplin, MO 64801
R. D. Connely

MONTANA

Powder River Computer Club
Powder River County High School
Broadus, MT 59317
Contact: Jim Sampson

Commodore User Club
1109 West Broadway
Butte, MT 59701
Contact: Mike McCarthy

NEBRASKA

Greater Omaha Commodore 64 Users Group
2932 Leawood Dr.
Omaha, NE 68123
(402) 292-2753
Bob Quisenberry

NEVADA

Las Vegas PET Users
Suite 5-315
5130 E. Charleston Blvd.
Las Vegas, NV 89122
Gerald Hasty

NEW JERSEY

Commodore Friendly User Group
49 Hershey Rd.
Wayne, NJ 07470
(201) 696-8043
Rich Pinto/Colin Campbell

Somerset Users Club
49 Marcy Street
Somerset, NJ 08873
Contact: Robert Holzer

Educators Advisory
P.O. Box 186
Medford, NJ 08055
(609) 953-1200
John Handfield

VIC-TIMES
46 Wayne Street
Edison, NJ 08817
Thomas R. Molnar

VIC 20 User Group
67 Distler Ave.
W. Caldwell, NJ 07006
(201) 284-2281
G. M. Amin

VIC Software Development Club
77 Fomalhaut Ave.
Sewell, NJ 08080
H. P. Rosenberg

ACGNJ PET/VIC/CBM User Group
30 Riverview Terr.
Belle Mead, NJ 08502
(201) 359-3862
J. M. Pylka

South Jersey Commodore Computer Users Club
46-B Monroe Park
Maple Shade, NJ 08052
(609) 667-9758
Mark Orthner
2nd Fri. of month

Parsippany Computer Group
51 Ferncliff Rd.
Morris Plains, NJ 07950
(201) 267-5231
Bob Searing

NEW HAMPSHIRE

Northern New England Computer Society
P.O. Box 69
Berlin, NH 03570

TBH VIC-NICs
P.O. Box 981
Salem, NH 03079

C-64 U.S.E.R.S. User Software Exchange Pro
P.O. Box 4022
Rochester, NH 03867
Paul Kyle

NEW MEXICO

Commodore Users Group
6212 Karlson, NE
Albuquerque, NM 87113
(505) 821-5812
Danny Byrne

NEW YORK

Capital District 64/VIC 20 Users Group
363 Hamilton St.
Albany, NY 12210
(518) 436-1190
Bill Pizer

Long Island PET Society
Ralph Bressler
Harborfields HS
Taylor Avenue
Greenlawn, NY 11740

PET User Club of Westchester
P.O. Box 1280
White Plains, NY 10602
Contact: Ben Meyer

LIVE (Long Island VIC Enthusiasts)
17 Picadilly Road
Great Neck, NY 11023
Contact: Arnold Friedman

Commodore Masters
25 Croton Ave.
Staten Island, NY 10301
Contact: Stephen Farkouh

VIC Users Club
76 Radford St.
Staten Island, NY 10314
Contact: Michael Frantz

West Chester County VIC Users Group
P.O. Box 146
Pelham, NY 10552
Joe Brown

SPUG
4782 Boston Post Rd.
Pelham, NY 10803
Paul Skipski

VIC 20 User Club
151-28 22nd Ave.
Whitestone, NY 11357
Jean F. Coppola

VIC 20 User Club
339 Park Ave.
Babylon, NY 11702
(516) 669-9126
Gary Overman

VIC User Group
1250 Ocean Ave.
Brooklyn, NY 11230
(212) 859-3030
Dr. Levitt

L&M Computer Club
VIC 20 & 64
4 Clinton St.
Tully, NY 13159
(315) 696-8904
Dick Mickelson

Commodore Users Group
1 Corwin Pl.
Lake Katrine, NY 12449
J. Richard Wright

VIC 20/Commodore 64 Users Group
31 Maple Dr.
Lindenhurst, NY 11757
(516) 957-1512
Pete Lobol

VIC Information Exchange Club
336 W. 23 St.
Deer Park, NY 11729
Tom Schlegel
SASE & phone please

New York Commodore Users Group
380 Riverside Dr., 7Q
New York, NY 10025
(212) 566-6250
Ben Tunkelang

Hudson Valley Commodore Club
1 Manor Dr.
Woodstock, NY 12498
F.S. Goh
1st Wednesday of month

LIVICS (Long Island VIC Society)
20 Spyglass Lane
East Setauket, NY 11733
(516) 751-7844
Lawrence Stefani

VIC Users Group
c/o Stoney Brook Learning Center
1424 Stoney Brook Rd.
Stoney Brook, NY 11790
(516) 751-1719
Robert Wurtzel

Poughkeepsie VIC User Group
2 Brooklands Farm Rd.
Poughkeepsie, NY 12601
(914) 462-4518
Joe Steinman

VIC 20 User Group
Paper Service Division
Kodak Park
Rochester, NY 14617
David Upham, Sr.

Manhattan 64
426 West 48th
New York, NY 10036
(212) 307-6519
Charles Honce

Adirondack Commodore 64 Users Group
205 Woodlawn Ave.
Saratoga Springs, NY
(518) 584-8960
Paul Klompas

Rockland County Commodore Users Group
P.O. Box 573
Nanuet, NY 10965
Ross Garber

New York 64 Users Group
222 Thompson St.
New York, NY 10012
(212) 673-7241
Bruce Cohen

Finger Lakes Commodore Users Group
c/o Rose City Computer Associates
229 West Union St.
Newark, NY 14513
(315) 331-1185

The Commodore Users Group Rochester
78 Hardison Rd.
Rochester, NY 14617
(716) 544-5251
Tom Werenski
Phone Evenings between 7-10

NORTH CAROLINA

Amateur Radio PET Users Group
P.O. Box 30694
Raleigh, NC 27622

Contact: Hank Roth

VIC Users Club
c/o David C. Fonenberry
Route 3, Box 351
Lincolnton, NC 28092

Microcomputer Users Club
Box 17142 Bethabara Sta.
Winston-Salem, NC 27116
Joel D. Brown

VIC Users Club
Rt. 11, Box 686
Hickory, NC 28601
Tim Gromlovits

Raleigh VIC 20/64 Users Group
410-D Delta Court
Cary, NC 27511
(919) 469-3862
Larry Diener

OHIO

Dayton Area PET
User Group
933 Livingston Drive
Xenia, OH 45385
B. Worby, President
(513) 848-2065
J. Watson, Secretary
(513) 372-2052

Central Ohio PET
Users Group
107 S. Westmoor Avenue
Columbus, OH 43204
(614) 274-0304
Contact: Philip H. Lynch

Commodore Computer Club of Toledo
734 Donna Drive
Temperance, MI 48182
Gerald Carter

Chillicothe Commodore
Users Group
P.O. Box 211
Chillicothe, OH 45601
William A. Chaney

Licking County 64 Users Group
323 Schuler St.
Newark, OH 43055
(614) 345-1327

11433 Pearl Rd.
Strongsville, OH 44136
Paul M. Warner

C.P.U. Connection
P.O. Box 42032
Brook Park, OH 44142
Danni Hudak

Commodore Users Group
18813 Harlan Dr.
Maple Heights, OH 44137
(216) 581-3099
Carl Skala

Commodore Users
of Blue Chip (Cincinnati)
816 Beecher St.
Cincinnati, OH 45206
(513) 961-6582
Ted Stalets

OKLAHOMA

Southwest Oklahoma
Computer Club
c/o Commodore Chapter
P.O. Box 6646
Lawton, OK 73504
1:30 at Lawton City Library

Tulsa Area Commodore Users Group
Tulsa Computer Society
P.O. Box 15238
Tulsa, OK 74112
Annette Hinshaw

Commodore Oklahoma Users Club
4000 NW 14th St.
Oklahoma City, OK 73107
(405) 943-1370
Stanley B. Dow

Commodore Users
Box 268
Oklahoma City, OK 73101
Monte Maker, President

Commodore Users of Norman
209 Brookwood
Noble, OK 73068
Matt Hager

OREGON

NW PET Users Group
John F. Jones
2134 N.E. 45th Avenue
Portland, OR 97213

Pacific Northwest Commodore
Users Group
P.O. Box 2310
Roseburg, OR 97470
(503) 672-7591
Richard Tsukiji

PENNSYLVANIA

PET User Group
Gene Beals
P.O. Box 371
Montgomeryville, PA 18936

Penn Conference Computer Club
c/o Penn Conference of SDA
720 Museum Road
Reading, PA 19611
Contact: Dan R. Knepp

PACS Commodore Users Group
LaSalle College
20th & Olney Ave.
Philadelphia, PA 19141
(215) 951-1258
Stephen Longo

Glen Schwartz
807 Avon
Philadelphia, PA 19116

Gene Planchak
4820 Anne Lane
Sharpsville, PA 15150
(412) 962-9682

PPG (Pittsburgh PET Group)
c/o Joel A. Casar, DMD
2015 Garrick Drive
Pittsburgh, PA 15235
(412) 371-2882

Westmoreland Commodore
Users Club
c/o DJ & Son Electronics
Colonial Plaza
Latrobe, PA 15650
Jim Mathers

COMPSTARS
440 Manatawny St.
Pottstown, PA 19464
Larry Shupinski, Jr.
Meet at Audio Video
Junction

Commodore Users Club
3021 Ben Venue Dr.
Greensburg, PA 15601
(412) 836-2224
Jim Mathers

VIC 20 Programmers, Inc.
c/o Watson Woods
115 Old Spring Rd.
Coatesville, PA 19320
Robert Gougher

G.R.C. User Club
300 Whitten Hollow Rd.
New Kensington, PA 15068
Bill Bolt

NADC Commodore Users Club
248 Oakdale Ave.
Horsham, PA 19044
Norman McCrary

CACC (Capitol Area Commodore
Club)
134 College Hill Rd.
Enola, PA 17025
(717) 732-2123
Lewis Buttery
Union Deposit Mall at 7 p.m.

G/C Computer Owners Group
c/o Gilbert Associates, Inc.
P.O. Box 1498
Reading, PA 19607
Extension 6472
Jo Lambert (215) 775-2600

Boeing Employees Personal
Computer Club
The Boeing Vertol Co.
P.O. Box 16858
Philadelphia, PA 19142
(215) 522-2257
Jim McLaughlin

South Central PA Commodore Club
2109 Cedar Run Dr.
Camp Hill, PA 17011
(717) 763-4219
David Persing

Main Line Commodore Users
Group (MLCUG)
c/o Main Line Computer Center
1046 General Allen Lane
West Chester, PA 19380
(215) 388-1581
Emil Volcheck

Commodore Users Group
781 Dick Ave.
Warminster, PA 18974
Matt Matulaitis

The Commodore Users Club
of S.E. Pittsburgh
c/o Groves Appliance & TV
2407 Pennsylvania Ave.
West Mifflin, PA 15122
Charles Groves

PUERTO RICO

CUG of Puerto Rico
RFD #1, Box 13
San Juan, PR 00914
Ken Burch

VIC 20 User Group
655 Hernandez St.
Miramar, PR 00907
Robert Morales, Jr.

RHODE ISLAND

Irving B. Silverman, CPA
160 Taunton Ave.
E. Providence, RI 02914
Contact: Michelle Chavanne

Newport VIC/64 Users
10 Maitland Ct.
Newport, RI 02840
(401) 849-2684
Dr. Matt McConeghy

The VIC 20 Users Club
Warwick, RI 02886
Tom Davey

Commodore Users Group
c/o Data-Co.
978 Tiogue Ave.
Coventry, RI 02816
(401) 828-7385
Victor Moffett

SOUTH CAROLINA

Beaufort Technical College
100 S. Ribaut Rd.
Beaufort, SC 29902
Dean of Instruction

Computer Users Society
of Greenville
Horizon Records-Home Computers
347 S. Pleasantburg Dr.
Greenville, SC 29607
(803) 235-7922
Bo Jeanes

Commodore Computer Club
of Columbia
318 Quincannon Dr.
Columbia, SC 29210
Buster White Sect./Treas.

Spartanburg Commodore Users Group
803 Lucerne Dr.
Spartanburg, SC 29302
(803) 582-5897
James Pasley

SOUTH DAKOTA

PET User Group
515 South Duff
Mitchell, SD 57301
(605) 996-8277
Contact: Jim Dallas

VIC/64 Users Club
608 West 5th
Pierre, SD 57501
(605) 224-4863
Larry Lundeen

TENNESSEE

River City Computer
Hobbyists
Memphis, TN
1st Mon. at Main Library

Nashville Commodore Users Group
P.O. Box 121282
Nashville, TN 37212
(615) 331-5408
Dave Rushing
3rd Thurs. at Cumberland Mus

Commodore User Club
Metro Computer Center
1800 Dayton Blvd.
Chattanooga, TN 37405
Mondays 7:30 pm

Metro-Knoxville 64 Users Club
7405 Oxmoor Rd., Rt. #20
Knoxville, TN 37921
(615) 938-3773
Ed Pritchard

Memphis Commodore Users Group
2476 Ridvers Ave.
Memphis, TN 38127

user groups

(901) 358-5823
Harry Ewart

TEXAS

PET Users
2001 Bryan Tower
Suite 3800
Dallas, TX 75201

Larry Williams
P.O. Box 652
San Antonio, TX 78293

PET User Group
John Bowen
Texas A & M
Microcomputer Club
Texas A & M, TX

CHUG (Commodore Houston Users Group)
8738 Wildforest
Houston, TX 77088
(713) 999-3650
Contact: John Walker

Corpus Christi Commodores
3650 Topeka St.
Corpus Christi, TX 78411
(512) 852-7665
Bob McKelvy

Commodore Users Group
5326 Cameron Rd.
Austin, TX 78723
(512) 459-1220
Dr. Jerry D. Frazee

VIC Users Group
3817 64th Dr.
Lubbock, TX 79413

Southeast Houston VIC Users Group
11423 Kirk Valley Dr.
Houston, TX 77089
(713) 481-6653

64 Users Group
2421 Midnight Circle
Plano, TX 75075
S.G. Grodin

Savid Computer Club
312 West Alabama
Suite 2
Houston, TX 77006
Davi Jordan, Chairman

Gulf Coast Commodore Users Group
P.O. Box 128
Corpus Christi, TX 78403
(512) 887-4577
Lawrence Hernandez

Mid-Cities Commodore Club
413 Chisolm Trail
Hurst, TX 76053
Garry Wordelman

Mid-Cities Commodore Club
413 Chisolm Trail
Hurst, TX 76053
Bruce Nelson

Interface Computer Club
814 North Sabinas
San Antonio, TX 78207
M. E. Garza, President

Gulf Coast Commodore Users Group
P.O. Box 128
Corpus Christi, TX 78403
(512) 887-4577
Lawrence Hernandez

UTAH

Utah PUG
Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

The Commodore Users Club
742 Taylor Avenue
Ogden, UT 84404
Contact: Todd Woods Kap, President
David J. Shreeve, Vice President

The VIClic
799 Ponderosa Drive
Sandy, UT 84070
Contact: Steve Graham

VIC 20 Users
324 N. 300 W.
Smithfield, UT 84335
Dave DeCorso

Northern Utah VIC & 64 Users Group
P.O. Box 533
Garland, UT 84312
David Sanders

The Commodore Users Group
652 West 700 North
Clearfield, UT 84015
(801) 776-3950
Rodney Keller, Richard Brenchly

VIRGINIA

Northern VA PET Users
Bob Karpen
2045 Eakins Court
Reston, VA 22091
(803) 860-9116

VIC Users Group
Rt. 2, Box 180
Lynchburg, VA 24501
Contact: Dick Rossignol

VIC Users Group
c/o Donnie L. Thompson
1502 Harvard Rd.
Richmond, VA 23226

Dale City Commodore User Group
P.O. Box 2004
Dale City, VA 22193
(703) 680-2270
James Hogler

Tidewater Commodore Users Group
4917 Westgrove Rd.
Virginia Beach, VA 23455
Fred Monson

Fredericksburg Area Computer Enthusiasts
P.O. Box 324
Locust Grove, VA 22508
(703) 972-7195
Michael Parker

Commonwealth 20/64 Users Group
1773 Wainwright Dr.
Reston, VA 22090
(703) 471-6325
Tal Carawan, Jr.

VIC 20 Victims
4301 Columbia Pike #410
Arlington, VA 22204
(703) 920-0513
Mike Spengel

Peninsula Commodore 64 Users Group
124 Burnham Place
Newport News, VA 23606
(804) 595-7315
Richard G. Wilmoth

Norfolk Users Group
1030 West 43rd St. B-4
Norfolk, VA 23508
489-8292
Larry Pearson

NASA VIC 20 User Group
713 York Warwick Dr.
Yorktown, VA 23692
Harris Hamilton

135 Beverley Rd.
Danville, VA 24541
David Gray

R.A.C.E. Commodore Users Group
4726 Horseman Dr.
Roanoke, VA 24019
(703) 362-3960
Larry Rackow

WASHINGTON

NW PET Users Group
2565 Dexter N. 3203
Seattle, WA 98109
Contact: Richard Ball

PET Users Group
c/o Kenneth Tong
1800 Taylor Ave. N102
Seattle, WA 98102

Whidbey Island Commodore Computer Club
947 N. Burroughs Ave.
Oak Harbor, WA 98277
Michael D. Clark

Central Washington Commodore Users Group
1222 S. 1st St.
Yakima, WA 98902
Tim McElroy

Blue Mountain Commodore Users Club
667 Canary Dr.
Walla Walla, WA 99362
(509) 525-5452
Keith Rodue

Spokane Commodore User Group
N. 4311 Whitehouse
Spokane, WA 99205
(509) 328-1464
Stan White

WEST VIRGINIA

Personal Computer Club
P.O. Box 1301
Charleston, WV 25325
Cam Cravens

TriState Commodore Users
73 Pine Hill Estates
Kenova, WV 25530
(304) 453-2124
Marc Hutton

WISCONSIN

Sewpus
c/o Theodore J. Polozynski
P.O. Box 21851
Milwaukee, WI 53221

Waukesha Area Commodore User Group (WACUG)
256½ W. Broadway
Waukesha, WI 53186
Contact: Walter Sadler
(414) 547-9391

Commodore User Group
1130 Elm Grove St.
Elm Grove, WI 53122
Tony Hunter

Commodore 64 Software Exchange Group
P.O. Box 224
Oregon, WI 53575
E. J. Rosenberg

C.L.U.B. 84
6156 Douglas Ave.
Caledonia, WI 53108
(414) 835-4645 pm
Jack White
2nd Sat every month 10:00 am

VIC-20 & 64 User Group
522 West Bergen Dr.
Milwaukee, WI 53217
(414) 476-8125
Mr. Wachtl

Menomonie Area Commodore Users Group
510 12th St.
Menomonie, WI 54751
(715) 235-4987
Mike Williams

C.U.S.S.H.
3614 Sovereign Dr.
Racine, WI 53406
(414) 554-0156
Tim Tremmel
3rd Saturday of month

Madison Area Commodore Users Group
1552 Park St.
Middleton, WI 53562
(608) 831-4852
John Carvin
3rd Thurs. each month

S.W.I.T.C.H.
W156 N8834 Pilgrim Rd.
Menomonee Falls, WI 53051
(414) 255-7044
Len Lutz

Milwaukee Area CBM64 Enthusiasts (M.A.C.E.)
P.O. Box 340
Elm Grove, WI 53122
(414) 259-5991
Kevin Wilde

The Eau Claire CBM64 Users Group
Rt. 5, Box 179A
Eau Claire, WI 54703
(715) 874-5972
John Slavsky, Jr.

WYOMING

Commodore Users Club
c/o Video Station
670 North 3rd #B
Laramie, WY 82070
(307) 721-5908
Pamela Nash

CANADA

Toronto PET Users Group, Inc.
1912A Avenue Rd., Ste. 1

Toronto, Ontario, Canada
M5M 4A1
(416) 782-8900
or call 416-782-9252
Contact: Chris Bennett

PET Users Club
c/o Mr. Brown
Valley Heights Secondary School
Box 159
Langton, Ont. N0E 1G0

Vancouver PET Users Group
P.O. Box 91164
West Vancouver, British Columbia
Canada V7V 3N6

CCCC (Canadian Commodore Computer Club)
c/o Strictly Commodore
47 Coachwood Place
Calgary, Alberta, Canada
T3H 1E1
Contact: Roger Olanson

W.P.U.G.
9-300 Enniskillen Ave.
Winnipeg, Manitoba R2V 0H9
Larry Neufeld

VIC-TIMS
2-830 Helena St.
Trail, British Columbia
V1R 3X2
(604) 368-9970
Greg Goss

Arva Hackers
Medway High School
Arva, Ontario N0M 1C0
D. Lerch

Nova Scotia Commodore Computer Users Group
66 Landrace Cres.
Dartmouth, N.S. B2W 2P9
Andrew Cornwall

Bonnyville VIC Cursors
Box 2100
Bonnyville, Alberta T0A 0L0
(403) 826-3992
Ed Wittchen

Commodore Users Club of Sudbury
938 Brookfield Ave.
Sudbury, Ontario
P3A 4K4

PET Educators Group
P.O. Box 454
Station A
Windsor, Ontario
N9A 6L7

COMVIC
P.O. Box 1688
St. Laurent
Montreal, Quebec
H4L 4Z2

Calgary Commodore Users Group
37 Castleridge Dr., N.E.
Calgary, Alberta
T3J 1P4
John Hazard

FINLAND

VIC-Club in Helsinki
c/o Matti Aarnio
Linnustajankj 2B7
SF-02940 ESP00 94
Finland

GERMANY

Kettenberg 24
D 5880 Lueden Scheid
West Germany
Rudi Ferrari

ITALY

Commodore 64 Club
Universita di Studi shan
V. Avigliana 13/1
10138 TORINO
ITALY

KOREA

Commodore Users Club
K.P.O. Box 1437
Seoul, Korea
Contact: S. K. Cha

MEXICO

Asociacion De Usarios Commodore
c/o Alejandro Lopez
Arechiga
Holbein 174-6° Piso
Mexico 18, D.F.

Club de Usarios Commodore
Sigma del Norte
Mol del Valle, Local 44
Garza Garcia, N.L. 66220

NEW ZEALAND

Commodore Users Group
Meet at VHF Clubrooms
Hazel Ave.
Mount Roskill
3rd Wed. of month, 7:30 pm
Roger Altena 278-5262

Nelson VIC Users Group
c/o P.O. Box 860
Nelson, New Zealand
Peter Archer

E.R. Kennedy
c/o New Zealand Synthetic Fuels Corp. Ltd.
Private Bag
New Plymouth

NORWAY

VIC Club of Norway
Nedre Bankegt 10,
1750 Halden
Norway

UNITED KINGDOM

North London Hobby Computer Club
Dept. of Electronics & Communications Engineering
The Polytechnic of North London
Holloway Rd.
London N7 8DB

Croydon Microcomputer Club
111 Selhurst R.
Selhurst, London SE25 6LH
01-653-3207
Vernon Gifford

User Bulletin Board

Telecommunications Bulletin Board For Commodore 64 Users:

(201) 521-2432
All day Sunday, Monday, Tuesday; 7 p.m.-10 a.m. Wednesday, Friday, Saturday; 9 p.m.-10 a.m. Thursday.

VIC 20 or Commodore 64 Newsletter Exchange:

If your user group would like to exchange newsletters, contact John Warren, Jr., 58 N. Ewing Street, Indianapolis, IN 46201.

User Group for Kids Forming:

Contact Mike J. Sluchinski
338 Costigan Crest
Saskatoon, Saskatchewan
Canada S7J 3P3
373-0957

Attention VIC 20/Commodore 64 Owners:

We'll type any program to tape for $5.00. Send listing or magazine pages for published programs, your tape and return postage to:
Tom Keough
Heartland User Group
1220 Bartow Road #23
Lakeland, FL 33801

that does not compute...

"Simons' BASIC"

Issue 26 (October/November)

In his article Jim Gracely said Simons' BASIC for the Commodore 64 is available on disk, which is not correct. It is available to users in the U.S. on cartridge.

"Self-Modifying Programs"

Issue 26 (October/November)

The program listing needs one correction in line 70. That line should read: 70 GOTO 130.

"Advanced Bit-Mapped Graphics on the Commodore 64, Part 2"

Issue 25 (August/September)

The program was written for use with a disk drive and was never checked using a datassette. Fortunately, the solution is simple. Just add this line to Listing #3 (Demonstration Program):

15 POKE 833,0

This will fix the problem some readers were having with the Draw command.

This is not really a mistake in the program, but rather, location 833 is assumed to be zero (which it is not if you load the program from datassette). The POKE 833,0 should always be used after loading a graphics program from tape, but before running it. Therefore, the best place for it is right at the beginning of the program.

that does not compute...

"Getting the Most Out of (and Into) Your Disk Drive, Part 3"

Issue 25 (August/September)

The final mailing list program as it was published in Issue 25 doesn't do much as it stands—unless you know how to tinker around with it. For those of you who don't know how to modify it, here is a completed version that works very well as a simple mailing list, for use with the 1541 disk drive.

```
10 REM ** DISPLAY MENU **
20 PRINT"[CLEAR]";:U=1
30 PRINT"[RVS,SPACE6]MENU[SPACE6,RVOFF]"
40 PRINT:PRINT:PRINT
50 PRINT"[RVS]1[RVOFF]...FORMAT[SPACE]DISKETTE"
60 PRINT
70 PRINT"[RVS]2[RVOFF]...NEW[SPACE]ITEM"
80 PRINT
90 PRINT"[RVS]3[RVOFF]...FIND[SPACE]ITEM"
100 PRINT:PRINT
110 PRINT"[RVS]4[RVOFF]...UPDATE[SPACE]ITEM"
120 PRINT:PRINT
130 PRINT"ENTER[SPACE]SELECTION[SPACE,SHFT R]"
140 GET A$:IF A$=""THEN 140
150 IF A$="1"THEN 200
160 IF A$="2"THEN 300
170 IF A$="3"THEN GOSUB 400
180 IF A$="4"THEN 500
190 GOTO 10
199 REM ** FORMAT A DISKETTE **
200 INPUT"DISKETTE[SPACE]NAME";D$
205 OPEN 15,8,15,"N:"+D$+",1A"
210 CLOSE 15
215 DATA A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,
    Y,Z
220 READ A$
225 OPEN 1,8,4,A$+",W"
230 PRINT#1,A$
235 CLOSE 1
240 IF A$="Z"THEN 20
245 GOTO 220
299 REM ** NEW ITEM **
300 REM ** NEW ITEM **
```

```
305 PRINT"[CLEAR]"
310 PRINT"[RVS,SPACE2]N[SPACE]E[SPACE]W[SPACE3]I[SPACE]T
    [SPACE]E[SPACE]M[SPACE3,RVOFF]"
315 PRINT:PRINT
320 PRINT"[RVS]1[RVOFF]...LAST[SPACE]NAME"
325 PRINT
330 PRINT"[RVS]2[RVOFF]...FIRST[SPACE]NAME"
335 PRINT
340 PRINT"[RVS]3[RVOFF]...STREET[SPACE]ADDRESS"
345 PRINT
350 PRINT"[RVS]4[RVOFF]...CITY"
355 PRINT
360 PRINT"[RVS]5[RVOFF]...STATE"
365 PRINT
370 PRINT"[RVS]6[RVOFF]...ZIP[SPACE]CODE"
375 PRINT
380 PRINT"[RVS]7[RVOFF]...SAVE"
385 PRINT
390 PRINT"[RVS]8[RVOFF]...EXIT"
392 PRINT:PRINT"ENTER[SPACE]SELECTION[SHFT R]":GOSUB 600
395 GOTO 305
399 REM ** FIND A FILE **
400 INPUT"FILE[SPACE]TO[SPACE]FIND";FI$
405 FR$=LEFT$(FI$,1)
410 OPEN 1,8,4,FR$+",R"
415 INPUT#1,G$
420 INPUT#1,G$,T,S
425 IF G$=FI$THEN 440
430 IF ST=0 THEN 420
435 PRINT ST:CLOSE 1:PRINT"FILE[SPACE]NOT[SPACE]FOUND"
    :FOR Z=1 TO 500:NEXT:S=1:RETURN
440 CLOSE 1
445 OPEN 15,8,15
450 OPEN 2,8,2,"#"
455 PRINT#15,"B-R:"2;0;T;S
460 FOR R=1 TO 6
465 INPUT#2,I$(R)
470 PRINT I$(R)
475 NEXT
477 PRINT"[DOWN2,RVS]PRESS[SPACE]SPACE[SPACE]BAR[SPACE]TO
    [SPACE]CONTINUE[RVOFF]"
478 GET A$:IF A$<>"[SPACE]"THEN 478
480 CLOSE 2:CLOSE 15:RETURN
```

that does not compute...

```
500 GOSUB 400:IF S=1 THEN S=0:GOTO 10
502 REM ** UPDATE ITEM **
505 PRINT"[CLEAR]"
510 PRINT"[RVS,SPACE2]U[SPACE]P[SPACE]D[SPACE]A[SPACE]T
    [SPACE]E[SPACE3]F[SPACE]I[SPACE]L[SPACE]E[SPACE3,
    RVOFF]"
515 PRINT:PRINT
520 PRINT"[RVS]1[RVOFF]...LAST[SPACE]NAME"
525 PRINT
530 PRINT"[RVS]2[RVOFF]...FIRST[SPACE]NAME"
535 PRINT
540 PRINT"[RVS]3[RVOFF]...STREET[SPACE]ADDRESS"
545 PRINT
550 PRINT"[RVS]4[RVOFF]...CITY"
555 PRINT
560 PRINT"[RVS]5[RVOFF]...STATE"
565 PRINT
570 PRINT"[RVS]6[RVOFF]...ZIP[SPACE]CODE"
575 PRINT
580 PRINT"[RVS]7[RVOFF]...SAVE"
585 PRINT
590 PRINT"[RVS]8[RVOFF]...EXIT"
592 PRINT:PRINT"ENTER[SPACE]SELECTION[SHFT R]":GOSUB 600
595 GOTO 505
597 GOSUB 800
599 GOTO 20
600 REM ** INPUT ROUTINE **
610 GET A$:IF A$=""THEN 610
620 IF VAL(A$)=0 THEN 600
630 A=VAL(A$)
640 IF A=7 THEN 700
650 IF A=8 THEN 20
660 PRINT"ITEM";A;
670 INPUT I$(A)
680 RETURN
700 REM ** SAVE ROUTINE **
710 OPEN 15,8,15,"IO":OPEN 2,8,4,"#"
715 IF U=1 THEN U=0:GOTO 760
720 PRINT#15,"B-A:"0;1;1
730 INPUT#15,A,B$,T,S
740 IF B$="OK"THEN T=1:S=1:GOTO 760
750 PRINT#15,"B-A:"0;T;S
760 FOR X=1 TO 6:PRINT#2,I$(X):NEXT
```

```
761 PRINT#15,"B-W:"4;0;T;S
770 CLOSE 2:CLOSE 15
780 F$=LEFT$(I$(1),1)
790 OPEN 1,8,4,F$+",R"
799 REM** SWAP THE CONTENTS **
800 OPEN 2,8,2,"TEMP,W"
810 INPUT#1,L$
815 SB=ST
820 PRINT#2,L$
825 IF SB<>0 THEN GOTO 840
830 GOTO 810
840 CLOSE 1
850 PRINT#2,I$(1)
860 PRINT#2,T
870 PRINT#2,S
880 CLOSE 2
890 OPEN 15,8,15,"S0:"+F$
900 PRINT#15,"R0:"+F$+"=TEMP"
910 CLOSE 15:RETURN
```

"Where Are We"

Issue 24 (June/July)

Our designers chopped some crucial spaces out of Liz Deal's machine language program listing here. The listing should look like this:

```
1 REM"S=SAVE"1:WHERE",08
500 REM------------------------------
510 REM WHERE ARE WE   ELIZABETH DEAL
520 REM------------------------------
530 A1=58:A2=62
540 DEFFNPP(Q)=PEEK(Q)+256*PEEK(Q+1)
550 DEFFNHI(Q)=INT(Q/256)
560 DEFFNLO(Q)=Q-256*FNHI(Q)
570 DATA NOT HERE
580 Y=FNPP(FNPP(A1)+1)-1:POKEA2,FNLO(Y):POKEA2+1,FNHI(Y)
590 REM READ POINTER HAS NOW BEEN SET TO THIS LINE
600 READQ$:PRINTQ$
610 DATA HERE
620 POKEY+6,94-PEEK(Y+6):REM NOW LIST
630 REM------------------------------
```

that does not compute...

```
640 :
650 DISASSEMBLY - $C2D7 CONTAINS RTS
660 ($60) IN UPGRADE PET
670 :
1100 .,  4080 20 D7 C2    JSR $C2D7
1101 .,  4083 BA          TSX
1102 .,  4084 BD FF 00    LDA $00FF,X
1103 .,  4087 A8          TAY
1104 .,  4088 BD 00 01    LDA $0100,X
1105 .,  408B 00          BRK
1106 .R
1107 :    PC   IRQ   SR AC XR YR SP
1108 .;  4080 9053 30 00 5E 04 F8
1109 .G
1110 B*
1111 :    PC   IRQ   SR AC XR YR SP
1112 .;  408C 9053 30 40 F8 82 F8
1113 .X                --    --
1114 READY.
1115 REM------------------------------
```

"Subject-Oriented Educational Software for Commodore Computers"

Issue 23 (April/May)

Throughout "Who Supplies What for Whom" on pages 63-64 we listed Courseware Magazine as a resource. That information is out of date. The company is now called School & Home Courseware, Inc. and is located at 1341 Bulldog Lane, Suite C, Fresno, California 93710. Phone 209-227-4341. C

new products

The following information is taken from new product announcements sent to us by independent manufacturers and is provided only to help keep our readers abreast of developments. Commodore does not endorse any of the products mentioned, has not tested them and cannot vouch for their availability. If you have any problems with any of the products listed here, please write to us.

Company:

Robot Shack
P.O. Box 582
El Toro, CA 92630
714-768-5798

Product:

Emotional Home Robots—Add "emotions" to the Robot Shack X-1 Robot kit or any robot built from scratch using Robot Shack parts. The "emotion" option con-

sists of a low-cost computer that mounts in the robot coupled to electronic devices that allow the robot to wobble with emotion, shriek with joy, turn red, shiver, etc., depending on circumstances. Includes software.
Price: Complete kits $99.95 and up. Information package available for $5.00 refundable with first order.

Company:
Swearingen Software
6312 W. Little York, Suite 197
Houston, TX 77088
713-937-6410

Product:
Pick That Tune—Game for the Commodore 64. Contains 100 different popular tunes divided into four categories. One to ten players bid on the number of notes they think they will need to identify a tune. Additional tune categories, each containing 100 songs, available separately.
Price: Base program with documentation $29.95 disk. Additional tunes $9.95 disk.

Company:
Powerline Software
P.O. Box 635
New Hartford, NY 13413

Product:
Casino Roulette—Game for the Commodore 64 or VIC 20 with 8K expander. Creates a roulette board display and places chips when bets are made. Allows for either European or American style play and for changing casino payoffs. Accepts up to five players at once. On tape for the VIC; on tape or disk for the 64.
Price: $19.95 any version.

Company:
Quality Data Services
2847 Waialae Avenue,
Suite 104
Honolulu, HI 96826
808-735-1202

Product:
COM-MASTER—Assembly language communications monitor for the SuperPET. Allows the SuperPET to emulate a Lear-Siegler ADM-3A at baud rates as high as 19,200. Also includes the ability to transmit and receive data files between a Commodore disk and a wide variety of remote systems. Fully buffered and interrupt-driven input and output through serial port, ASCII control codes via use of the OFF/RVS key as a CONTROL key, true serial BREAK key using the STOP key and more.
Price: $95.00

Company:
Spinnaker Software
215 First Street
Cambridge, MA 02142
617-868-4700

Product:
Cosmic Life—Game on cartridge for the Commodore 64, in the tradition of "Go!" and checkers. Available by Christmas.
Price: Contact company.

Company:
Sirius Software, Inc.
10364 Rockingham Drive
Sacramento, CA 95827
916-366-1195

Product:
Games for the Commodore 64—*Gruds in Space* is an adventure game with over 150 animated screens and a humorous story line. Another graphic adventure, *Blade of Blackpoole,* is also available. *Squish 'Em* is available on cassette for the VIC 20 with 8K expander as well as the 64. *Capture the Flag* was shown for the first time at Summer CES, and should be available soon.
Price: 64 versions $39.95 each. *Squish 'Em* for VIC $19.95.

Company:
Infinity Software
536 Curie Drive
San Jose, CA 95123
408-629-6208

Product:
Two adventure programs for the Commodore 64—*Bandits at 4 O'Clock,* on tape or disk, simulates the action of a B-36 tail gunner in high-res graphics. Requires joystick. *Stardate 6000,* which employs inter-active logic features, is a futuristic adventure on disk or tape. Uses keyboard.
Price: $29.00 each

Company:
Creative Software
230 East Caribbean Drive
Sunnyvale, CA 94089
408-745-1655

Product:
In the Chips—Educational cartridge game for the VIC 20. Tests

new products

the entrepreneurial abilities of one or two players, who develop and operate rival software companies. The object is to use your capital in the most efficient way in order to "out-profit" your competition. When players finish, they see their financial results for the quarter, and must use that information to make future inventory, pricing and budgeting decisions.
Price: $29.95

Company:
RAK Electronics
P.O. Box 1585
Orange Park, FL 32067-1585
904-264-6777

Product:
Formulas II—Software for ham radio operators with Commodore 64 or VIC 20 and 8K expander. Includes formulas for decimal/hex/binary conversion, decibel gain/loss, resistor color codes, coaxial cable loss, coil winding calculator and more.
Price: $6.95 plus $2.00 shipping.

Company:
Celestial Software
3010 Warrington Avenue
Lakeland, FL 33803
813-686-3311

Product:
Astronomy Pac—On cassette for Commodore 64 or VIC 20 with 8K expander. Generates its own almanac data and contains information on Aries and 57 of the most prominent stars, which enables the user to print the location of these stars for anytime from 1983 to 2000 from any location on earth. Tell the computer the time, date and your approximate position. Input the height and azimuth of the star. The computer then determines which star you observed. Package includes software, instructions, astrolabe, compass and book titled *The Stars* by H.A. Rey.
Price: $59.95

Company:
Computer Associates, Inc.
1304 13th Avenue S.
Fargo, ND 58103
701-280-1837

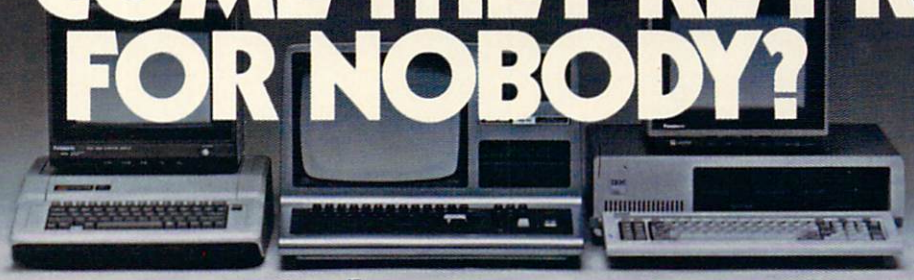
Product:
Video*Clear Interference Rejection Cable—Designed to eliminate or reduce video interference in microcomputers. Replaces the existing video cable between the computer and the T.V. set or monitor. Completely external.
Price: $14.95

Company:
bitCards
120 South University Drive, Suite F
Plantation, FL 33317
305-473-4741

Product:
bitCards—For the Commodore 64 or VIC 20. Text-and-graphics adventures intended to be given as gifts. Each bitCard is custom-programmed with personalized references to the recipient. The first, *A Christmas Adventure,* also contains a personal holiday greeting in the sender's own words.
Price: $18.50 all versions. C

advertisers' index

IF PERSONAL COMPUTERS ARE FOR EVERYBODY, HOW COME THEY'RE PRICED FOR NOBODY?



$1395* APPLE® IIe 64K

$999* TRS-80® III 16K

$1355* IBM® PC 64K

A personal computer is supposed to be a computer for persons. Not just wealthy persons. Or whiz-kid persons. Or privileged persons.

But person persons.

In other words, all the persons whom Apple, IBM, and Radio Shack seem to have forgotten about (including, most likely, you).

But that's okay. Because now you can get a high-powered home computer without taking out a second mortgage on your home.

It's the Commodore 64. We're not talking about a low-priced computer that can barely retain a phone number. We're talking about a memory of 64K. Which means it can perform tasks most other home computers can't. Including some of those that cost a lot more. (Take another look at the three computers above.)

By itself, the Commodore 64 is all the computer you'll ever need. Yet, if you do want to expand its capabilities some day, you can do so by adding a full complement of Commodore peripherals. Such as disk drives. Modems. And printers.

You can also play terrific games on the Commodore 64. Many of which will be far more challenging than those you could ever play on a game machine alone.

And as great as all this sounds, what's even greater-sounding is the price. It's hundreds of dollars less than that of our nearest competitor.

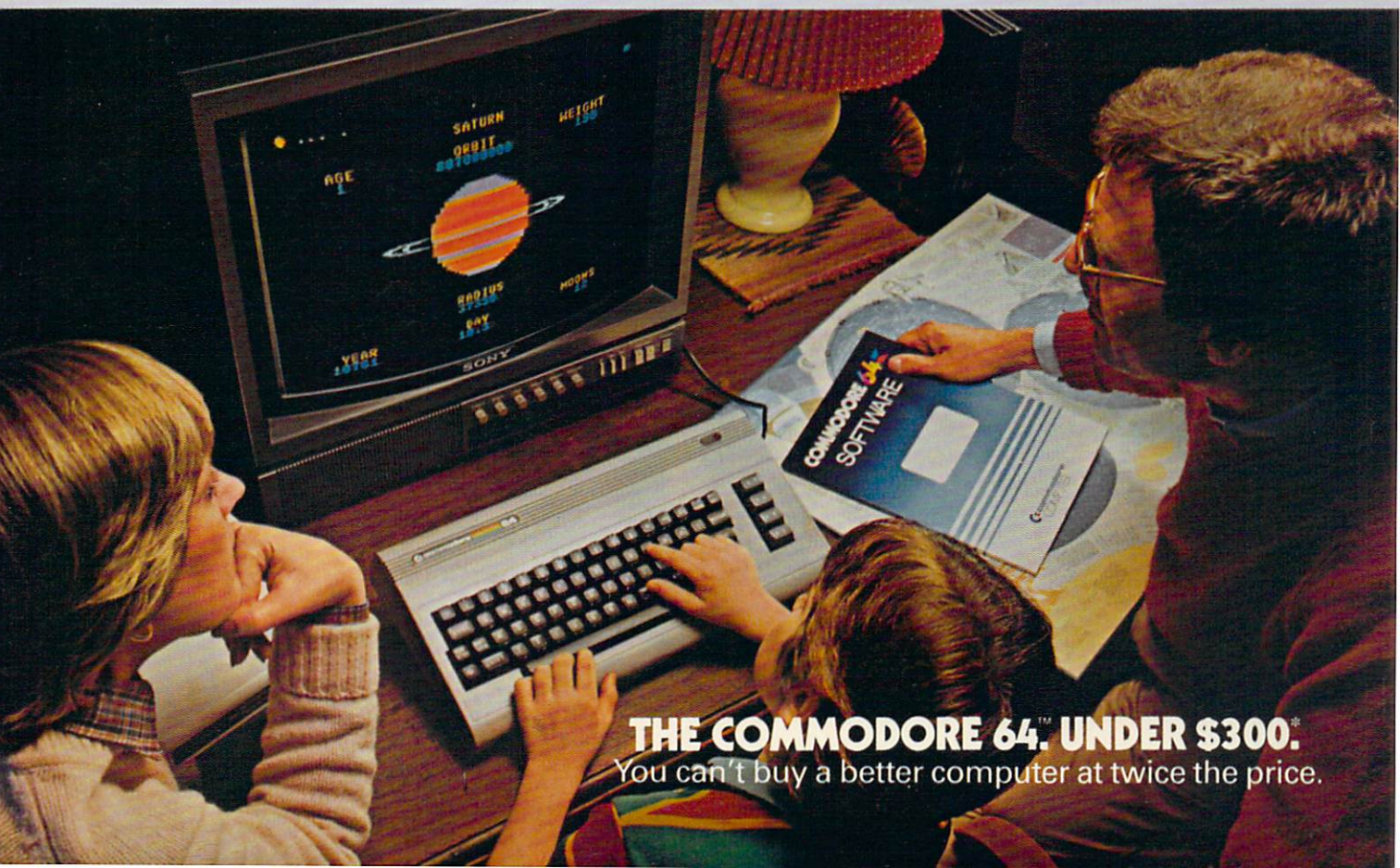
So while other companies are trying to take advantage of the computer revolution, it seems to us they're really taking advantage of something else:

Their customers.

*Manufacturers' suggested list prices. Monitor included with TRS-80 III only. Commodore Business Machines–P.O. Box 500R, Conshohocken, PA 19428; Canada–3370 Pharmacy Avenue, Agincourt, Ont., Can. M1W 2K4.

commodore COMPUTERS

Apple is a registered trademark of Apple Computer, Inc.
TRS-80 is a registered trademark of Tandy Corp. IBM is a registered trademark of International Business Machines Corp.



THE COMMODORE 64™. UNDER $300.*

You can't buy a better computer at twice the price.

ISBN 0-88731-005-2